Camera-In-The-Loop Test Bench Development for Advanced Driver Assistance Systems Applications

Undergraduate Honors Thesis

Timothy J. Kirby

Dr. Shawn Midlam-Mohler, Advisor

Spring, 2019

Department of Mechanical and Aerospace Engineering

The Ohio State University

This project is supported in large part by the Ohio State EcoCAR Student Project Team

Abstract

As the prevalence of ADAS and autonomous vehicles rises, the demands placed on the sensors and controllers they rely on also increases. To accommodate for this increase in demand, more robust validation methods will be required to enable reliable, fast, and comprehensive sensor and control process testing. For the camera sensor, one of the sensors used in many ADAS systems, a hardware-in-the-loop validation method, camera-in-the-loop (CIL), is useful as it allows for the camera sensor to be tested in real-world scenarios while also generating data to be used in the verification of the controller. However, taking live video from inside vehicles can be a costly and slow process that hinders the development of autonomous systems. Instead, if the environment that the camera sensor observes is simulated virtually using a high-fidelity modeler, similar results can be achieved in a less costly and more timely fashion. Using software would also allow for the rapid generation of new, specified test cases allowing for greater control over algorithm testing. In this project, a Mobileye 6, a camera sensor equipped with computer vision is mounted on a test bench and faces a digital display. The display visualizes an environment simulated in PreScan that shows a realistic roadway from the view of the camera. The multiple components in this test bench are then connected through a CAN bus that sends vehicle speed characteristics. In doing so, the Mobileye's computer vision interprets the displayed images and outputs valuable information such as object classification, object orientation, and various ADAS functions like lane departure warnings with a high degree of accuracy. Additionally, modifications to the test cases can be made in very little time while keeping the accuracy of the results consistent.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the countless groups and individuals that have made contributions to my work. First and foremost, I would like to thank my research advisor, Dr. Shawn Midlam-Mohler, for his supervision and insights into my project. He has constantly served as a level head to keep my research on course. When he first placed his trust in me and made me one of his researchers in 2017, I never could have imagined how far it would take me. Thanks to him, I have found an avenue through which to pursue my graduate degree and am exceptionally prepared for the career ahead of me. Additionally, he has served as an example from which I have molded my professional and personal goals. No matter the topic, Dr. Midlam-Mohler has been able to provide valuable information while maintaining his contagious smile and sense of humor. I will always be full of nothing but respect and appreciation for his help and aspire to be like him in numerous ways.

Additionally, I would like to thank Dr. Lisa Fiorentini for serving as a member of my defense committee. Her reputation throughout the department and Center for Automotive Research speaks to her expertise in control systems and I am privileged to have her join the committee. Her insights into the practical application of this system are essential.

A massive thanks must go to members of the Ohio State EcoCAR team. Evan and Simon, I have looked up to both of you since I joined the team in 2017. You are humble leaders who know so much more than you let on and are both surely bound for greatness. In fact, Evan, your constant willingness to educate me and connect me to further resources was essential to the completion of my work. Shlok, you and I collaborating on the debugging of the Mobileye and CAN communication helped me through the most challenging parts of my research. Kerri, you always went out of your way to make any wired

connection needed for the completion of this project, even in a pinch. To all of you and everyone else in EcoCAR, thank you for making this the experience what it was.

Lastly, I must thank my friends, family, and girlfriend. I never could have finished this project without your endless support. I leaned on you when times were at their toughest and I can only hope to be as kind to you as you have been to me.

1 Introduction

1.1 Background

Automation has quickly become an essential component of the automotive industry. In fact, 86% of all production vehicles in 2016 were equipped with some sort of advanced driver assistance system, ADAS (Ward's Intelligence, 2017). However, this trend shows no sign of slowing down. According to the Victoria Transport Policy Institute, autonomous vehicles will make up 50 percent of the vehicle sales by 2050 (Litman, 2019). The shift in the automotive standards is for good reason, though. Currently, 94% of accidents occur as a result of human error. Therefore, by automating driving and removing the human element, it can be assumed that the vast majority of accidents would be prevented. Which, when the United States' 39,141 deaths resulting from automotive accidents are put into perspective, over 35,000 lives could be saved per year (U.S. Department of Transportation, 2018).

The life saving features offered by autonomous vehicles rely on a wide array of sensors. One of the sensors required is the camera, which achieves its fullest functionality when it is partnered with a process called computer vision. Computer vision is the capability of a computer to use a two-dimensional image or a video and calculate relevant three-dimensional data from it. In automotive applications, cameras enabled with computer vision can output information such as obstacle detection, object classification, and vehicle trajectory, all of which are vital to the algorithms used in automated vehicle control (Ballard, 1982).

However, before these features can be distributed to consumers, they must be validated. Hardware-in-the-loop (HIL) is one such method of doing so. HIL is the use of the component or controller that requires validation in a realistic situation such that its response to the stimulus can be

observed (Ledin, 1999). For this project, because the camera sensor is the hardware in question, a camera-in-the-loop (CIL) will be manufactured. Additionally, to help validate computer vision software and ADAS features, the data from the CIL test bench can be output in order to assess the functionality of control processes.

Currently, camera-in-the-loop systems exist primarily for the two reasons listed: testing the camera and testing controllers that use the camera's data. To accomplish this, concepts of cameras observing simulations and outputting data have been generated previously (US Patent No. US7768527B2, 2006). For instance, tactical missile sensors have used this method for years (Ledin, 1999). Other aerospace applications have also been put to practice. Hardware-in-the-loop visual systems and controllers are being used by UAVs to remotely control their travel. However, this project relies on the use of an actual vehicle to record the data, which is both expensive and more dangerous than a simulated system (Prabowo, 2015). In an effort to avoid these concerns and allow for rapid redesign of test cases, a Gruyer et al. developed a similar system that points a UAV camera at a display and successfully simulates the environmental conditions of flying through the air. This is accomplished with multiple displays, custom made screen distortion algorithm to accommodate for the use of those multiple flat screens, and MultiGen-Paradigm's Vega Prime as the virtual reality simulator (Gans, 2009).

Other projects focus on smaller portions of the camera and controller testing. For instance, a project by Muller et al. focused on the generation of simulated environments with the intention of replacing on-road testing for ADAS systems. However, this project is still reliant on some visuals obtained from actual recordings and applies modifications retroactively from there (Muller, 2015). While this is a great improvement on a video being recorded for each test case, it still does not offer complete freedom. Similarly, Zhou et al. generates virtual test cases from GPS and radar data from test drives (Zhou, 2016). And while this is useful in the development of simulated environments, it once again is still

reliant on test drives in a physical vehicle, a hindrance to the development of test cases. However, the simulation develop by Gruyer et al. does not have that limitation. In the project, test cases for ADAS applications are generated as models and filters to make them more realistic are then applied. This is accomplished with ProSiVIC virtual platform. Being able to produce simulations with a similar level of control and detail would allow for excellent testing of the camera sensor (Gruyer, 2012).

Another vital component of modern vehicles is the controller area network (CAN). CAN is a connection of multiple controllers to send messages from component to component, also known as a CAN bus. Each message is assigned IDs in order of priority to avoid losing the most important messages or creating conflicting data. Because this system allows for messages to communicate without losing the most important ones, it is also capable of handling many messages and serving as the intermediate between all the controllers in a system. However, to prevent every message from reaching every controller, CAN networks also utilize a data acknowledgement process. Whenever a controller receives a message on the CAN bus, it must check the message and accept the data, thereby acknowledging back to the system that it has received it. Nowadays, a CAN bus is present in most modern vehicles, meaning that any system that wants to observe the messages being sent throughout the vehicle can simply attach to the CAN network and start observing the messages (CSS Electronics, 2019).

1.2 Focus of Research

This research was tasked with the development and validation of a camera-in-the-loop test bench capable of testing control algorithms and the functionality of camera sensors without the need to implement it in a vehicle. This was accomplished by directing a camera sensor at a digital display that renders realistic driving scenarios from a computer. Then, the information from the camera sensor's computer vision system was interpreted and output for the use in control algorithms. The CIL test bench

emulates the real-world using simulation software, allowing it to quickly generate test scenarios to target specific functions in an advanced driver assistance system. Additionally, it includes a CAN bus to emulate vehicle messaging that is required by the camera sensor. The camera sensor, a Mobileye 6, uses the vehicle information and digitally rendered test cases to output relevant warnings, detections, and obstacle information to be interpreted by control algorithms in the future.

1.3 Significance

As stated previously, ADAS is already prevalent and autonomous vehicles are anticipated to dramatically increase in popularity in the coming years (Litman, 2019). However, for these features to be implemented, they must also be validated. The validation process for camera sensors could be accomplished with hardware-in-the-loop systems. But, if the validation process is coupled with computer vision to generate test scenario data, the camera-in-the-loop (CIL) could also serve as a means of testing controllers. Moreover, by generating the scenarios for this testing platform virtually using PreScan, the camera sensor and controllers can be tested very quickly. Building a PreScan model is simple to do and allows for a high degree of freedom over the simulated environment. This gives the user the ability to test very specific test cases that might otherwise be dangerous in a short amount of time. For instance, if the user wanted to test a camera's ability to detect a high-speed stop and the controller's ability to do so without collision, they previously might have needed to physically drive a vehicle equipped with the sensor and put it in that dangerous situation. With this test bench, the physical testing can be avoided all together. A new test case could be generated and used in this CIL test bench in minutes without endangering the user or costing someone a vehicle.

1.4 Overview of Thesis

This thesis contains seven chapters. Chapter 2 elaborates on the selection process for the hardware used in the creation of the CIL test bench. Chapter 3 focuses on the development of a virtual environment that is capable of simulating real-world situations that would test the functionality of a camera sensor and the vehicle control algorithms that rely on it. Chapter 4 describes the calibration process used in the CIL and covers the advantages and disadvantages of the contending calibration methods. Chapter 5 details the

2 Hardware Selection and Development

One of the most important aspects of this project is selecting appropriate hardware to perform as a camera-in-the-loop system. This is because many of the parts are reliant on one another to satisfy their requirements. So, all necessary accommodations must be made for each component to function as intended.

2.1 Mobileye 6

The camera sensor is the most important component of a camera-in-the-loop test bench as it dictates the requirements of the rest of the system. For instance, if the digital display that the camera sensor is observing has lower resolution than the selected camera unit, precision is lost due to the loss in resolution of the interpreted images. Therefore, all other components of the CIL should be equal to or greater than the precision of the camera unit.

The Mobileye 6 is a windshield mounted camera sensor that enables advanced driver assistance systems in any vehicle that it is installed in [Figure 1]. It does so by using its camera, CAN messages from the vehicle it is in, and its own onboard computer to output warnings such as lane departure, brake

warnings, and traffic sign detection. The warnings are presented to the driver of the vehicle on a small display unit called the EyeWatch [Figure 2].

Figure 1: Mobileye 6 Camera Sensor

Figure 2: Mobileye EyeWatch Unit (Mobileye, 2016)

Table 1: Mobileye 6 Connections

Wire Name & Function	Wire color	Connector	Connection To
EyeCAN - (6 pin connector)	Black	P2	EyeCAN unit (for system calibration)
EyeWatch - (4 pin connector)	Black	J1	EyeWatch Display & Control unit
BAT+ (12/24V)	Red	-	Vehicle constant power (Battery)
GND	Black	-	Vehicle GND (BAT-)
Ignition (12/24V)	Blue	-	Vehicle Ignition signal
CAN B H	White	-	Vehicle CAN-bus (CAN High wire)
CAN B L	Yellow	-	Vehicle CAN-bus (CAN Low wire)
IHC – **(Analog Output)**	Gray	-	Vehicle High-beams via external Relay or connection to any 3rd party device
AUX (Analog Input)	Pink	-	*1 analog Signal Input (or both Left and Right Turn indicators analog input via Diode)*

The Mobileye 6 was selected as the camera sensor of choice for various reasons. Most importantly, it houses an onboard computer that interprets the camera view with Mobileye's computer vision systems. This allows users of the CIL test bench to avoid needing to develop their own computer vision, saving resources otherwise used to develop this process and thereby minimizing the barrier to entry for the testing of camera-based control methods. Additionally, Mobileye is a trusted brand for camera sensors and computer vision that is installed in more than 25 million vehicles and is ISO 9001, an international standardized measure of quality control, certified for the "development and manufacturing of video stream analysis systems" (Institute of Quality Control, 2018) (International Organization for Standardization, n.d.) (Mobileye, An Intel Company, n.d.). Based on this, it can be assumed that their computer vision process is robust and accurate, likely producing comparable or better results than a user-developed computer vision system.

However, by selecting the Mobileye 6, several important requirements were introduced for the CIL. First, a Mobileye must connect to an already existing CAN bus in order to function. But, because the intention of the camera-in-the-loop test bench is to avoid using a physical vehicle, a CAN bus network must be spoofed to send the required CAN messages, speed and turn signals, to the Mobileye. Therefore, a means of CAN communication is needed in the test bench. Additionally, the Mobileye 6 utilizes an Aptina MT9V024/D camera sensor, which has an active pixel resolution of 640 x 480, a 4:3 ratio. Therefore, the digital display must be greater in resolution (Mobileye, 2016). Additionally, the sensor has a refresh rate of 60 frames per second (fps), necessitating the display also have a refresh rate of 60 Hz or more (Semiconductor Components Industries, LLC., November). Next, the sensor has a horizontal field of view of 38 degrees and should therefore be located a distance from the digital display such that its horizontal view does not extend beyond the display's boundaries. This requirement will be addressed in the "Digital Display" section. The process of installing the Mobileye is shown in Figure 3.

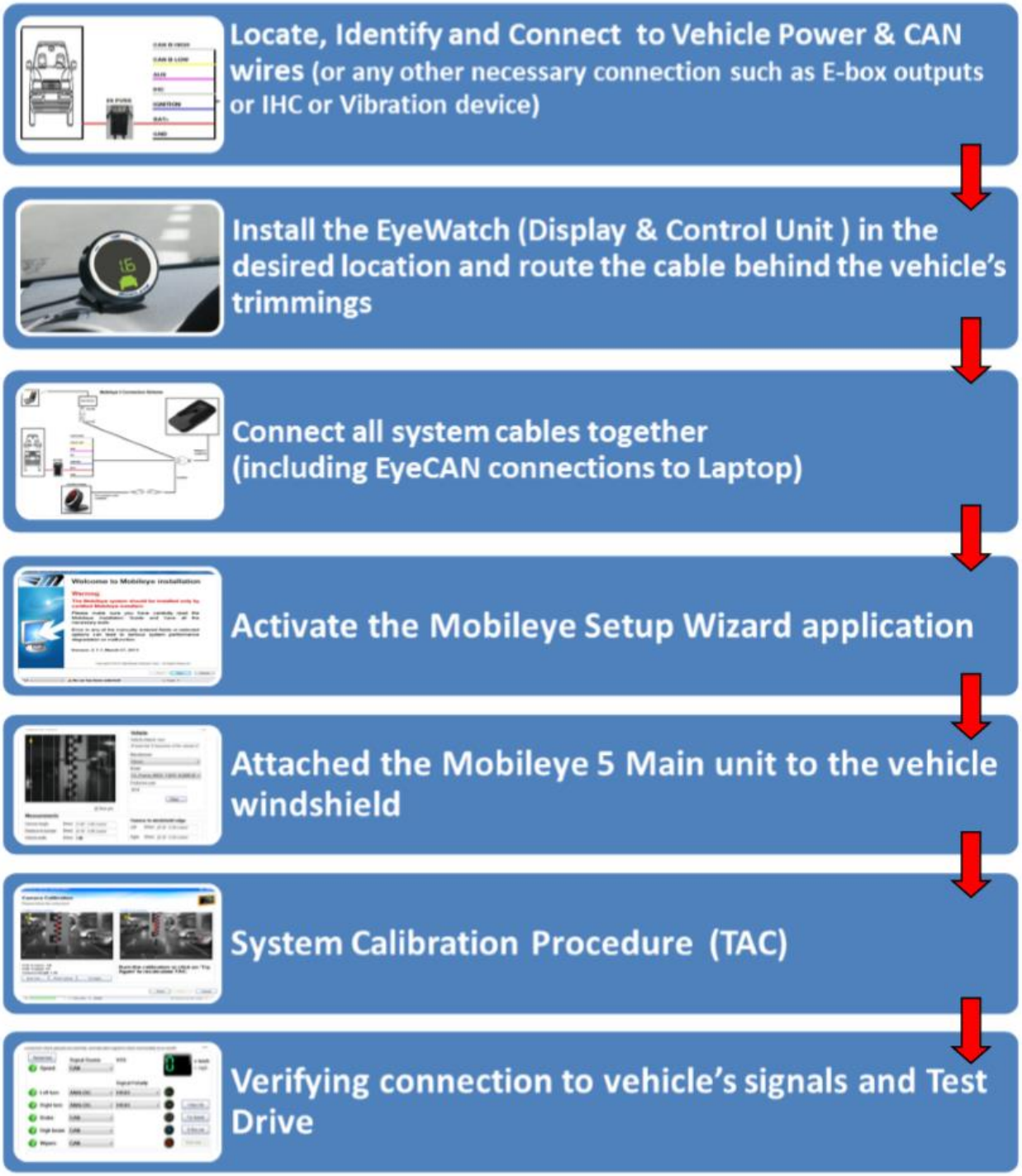

Figure 3: Mobileye 6 Installation Process (Mobileye, 2016)

2.2 CAN Communication

In order to recreate a CAN-bus, a minimum of two control units is usually required to communicate. The first unit is needed to prepare and send the CAN messages containing the information about the vehicle. The second device is needed to acknowledge the CAN messages.

To send the CAN messages, a CANcaseXL was selected. The team was already familiar with using the CANcaseXL, had one available, the CANcaseXL is capable of both sending and receiving, and it is capable of sending error messages, making it a valuable tool for debugging. To accept the messages, a preexisting Arduino controller that had been modified with an Arduino CAN bus module was chosen as the CAN receiver. The Arduino is far cheaper than a CANcaseXL and the CANcaseXL units had limited availability in lab. Together, they offered the flexibility of troubleshooting the CAN bus while remaining mindful of cost to the user, an objective of this project. The two control units were connected via a DB9 connection [Figure 4].

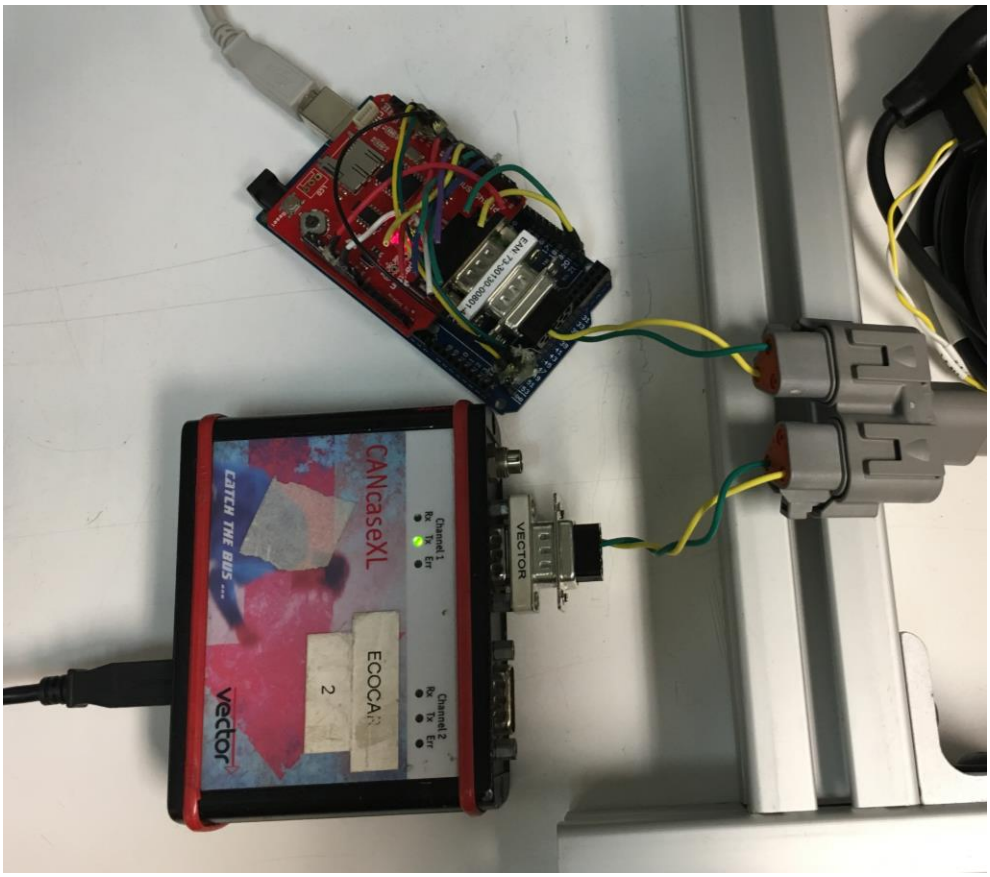

Figure 4: CAN Bus Network Emulating a Vehicle in The CIL Test Bench

2.3 Digital Display

The digital display selected for this project was a Sony LS27B350HS/ZA as it was available at the research facility and meets all requirements dictated by the Mobileye unit. It has a 27-inch screen with a resolution of 1920 x 1080 and a refresh rate of 60 Hz (Samsung, n.d.). Although, due to the 4:3 ratio used by the Mobileye, the only viewable pixels of the monitor would be a 1440 x 1080 (4:3) section at the center of the screen. Therefore, using simple geometry, the monitor dimensions, and the field of view of the Mobileye, the ideal location of the camera sensor relative to the monitor can be determined.

$$\sqrt{1920\ pixels^2 + 1080\ pixels^2} = Diagonal\ Pixels = 2202.91\ pixels$$

$$2202.91\ pixels = 27\ inches \qquad therefore \qquad 1440\ pixels = 17.65\ inches$$

$$\tan(angle) = \frac{opposite}{adjacent} = \tan\left(\frac{38}{2}\right) = \frac{\frac{17.65\ inches}{2}}{Distance\ from\ Monitor}$$

$$Distance\ from\ Monitor = 25.63\ inches$$

The monitor is also 15.7 inches tall with a stand that is 3.1 inches tall. Therefore, the center of the screen and the height of the camera sensor can be calculated.

$$Height = \frac{Monitor\ Height}{2} + Stand\ Height = \frac{15.7\ inches}{2} + 3.1\ inches = 10.95\ inches$$

2.4 Camera Mount

Following the determination of the height of the camera sensor, a mount was required that could hold the camera at least 11 inches in the air, replicate any visual side effects of a windshield, and hold the camera sensor at an angle similar to the mounting angle of an actual windshield, roughly 45 degrees. A rough CAD model was created to visualize this product [Figure 5]. The actual camera mount closely follows this intended design [Figure 6 and Figure 7].

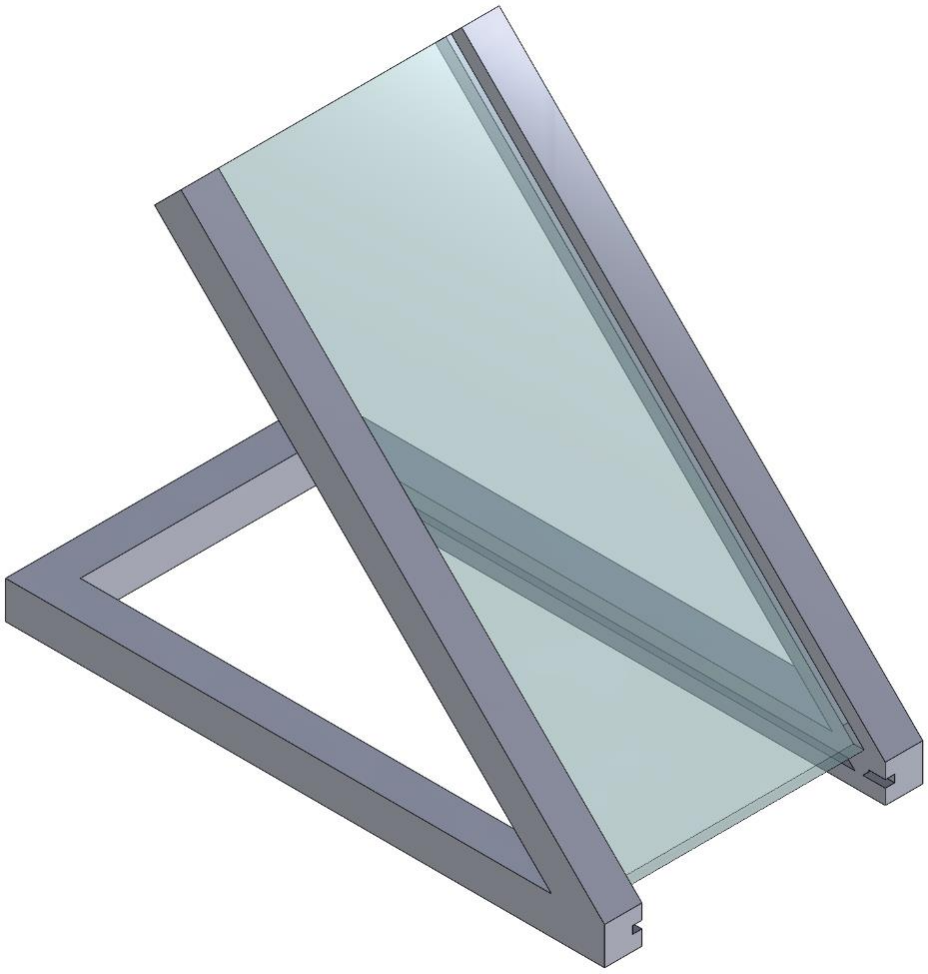

Figure 5: CAD Model of Proposed Camera Mount

Figure 6: First Version of Camera Mount

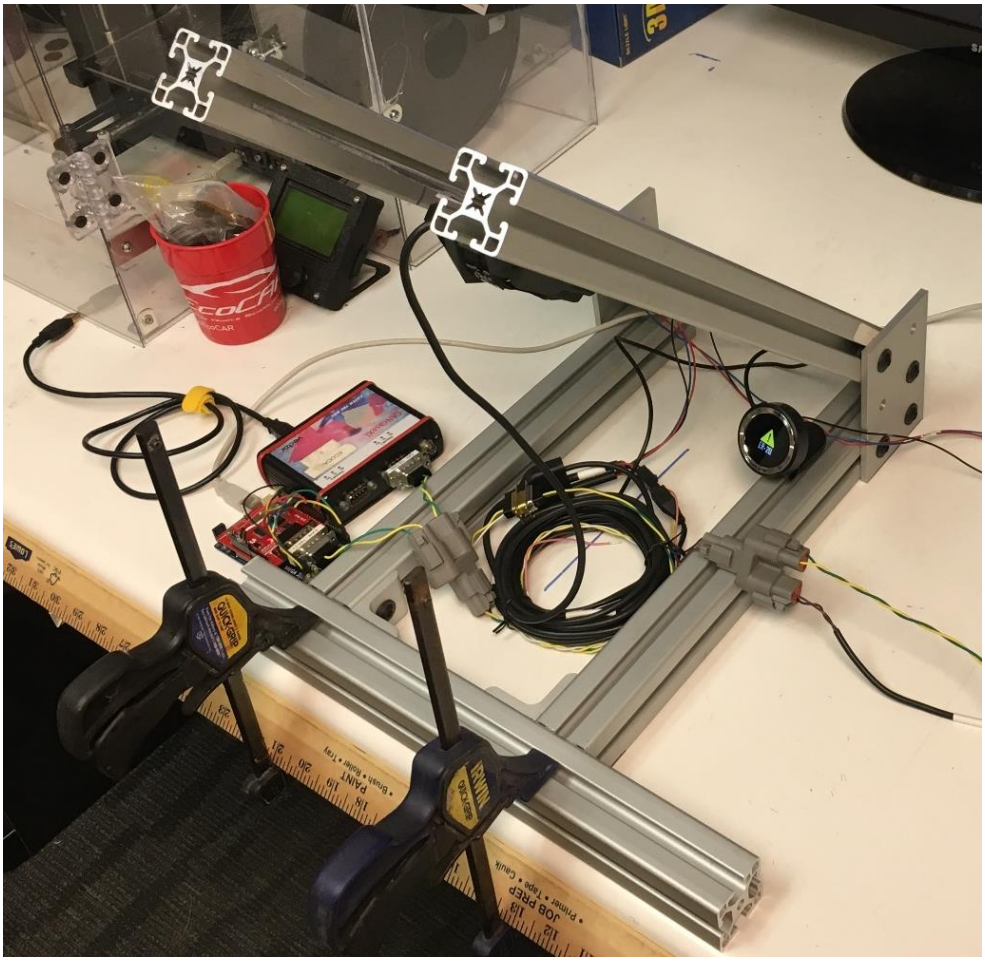

Figure 7: Final Version of Camera Mount

The final mount is made of 80/20 T-slotted aluminum, T-Nuts, and bolts. Then, a sheet of plexiglass was cut and mounted in the railing between the two angled members in order to serve as an artificial windshield to mount to [Figure 7].

2.5 Final Implementation

Lastly, several custom wire harnesses and custom connections required manufacturing in order to connect the various components of the CIL. For all pin connections, DB9 CAN bus connectors were

selected as they are standard for CAN connections. To connect the Mobileye to the CAN bus, the CAN B cord of the Mobileye unit, where CAN messages are received, had to be outfitted with a DB9 [Figure 8].

Figure 8: Mobileye 6 Connection to CAN Bus Outfitted with DB9 Connector

Additionally, the CAN connection between the CANcaseXL and the Arduino CAN module required a third output so the Mobileye 6 could receive the messages they sent. The three DB9 connections were connected using a DEUTSCH CAN network splitter. The 3-way DB9 harness can be seen below in Figure 9.

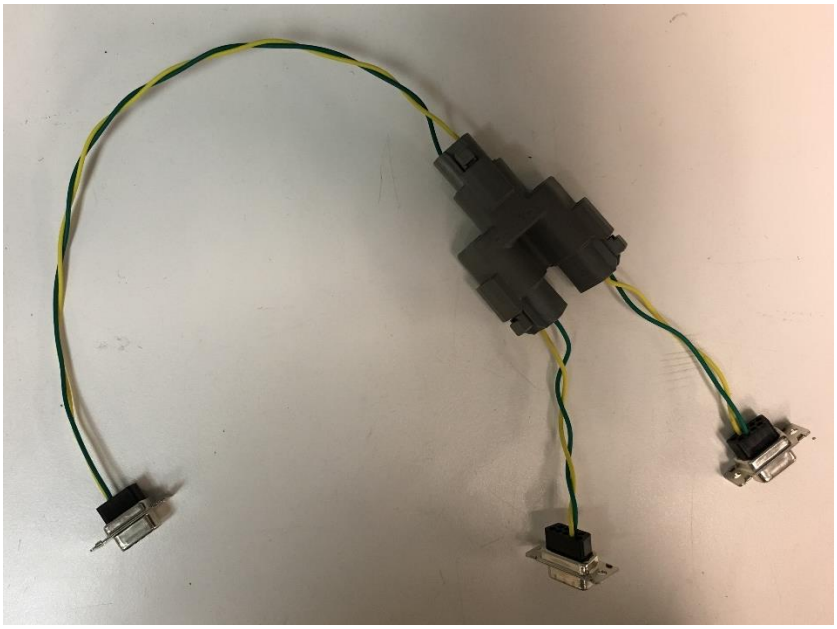

Figure 9: CAN Bus Network Splitter for Mobileye Connection

Additionally, the EyeCAN logging cable of the Mobileye 6 unit was required to record the data according to Mobileye documentation. However, the CAN-A cable came with a 6-pin connector that was used during the Mobileye calibration process. So, to enable logging, the EyeCAN cable was also split with a DEUTSCH connector. This left the 6-pin available for calibration but also allowed the Mobileye to record data [Figure 10].

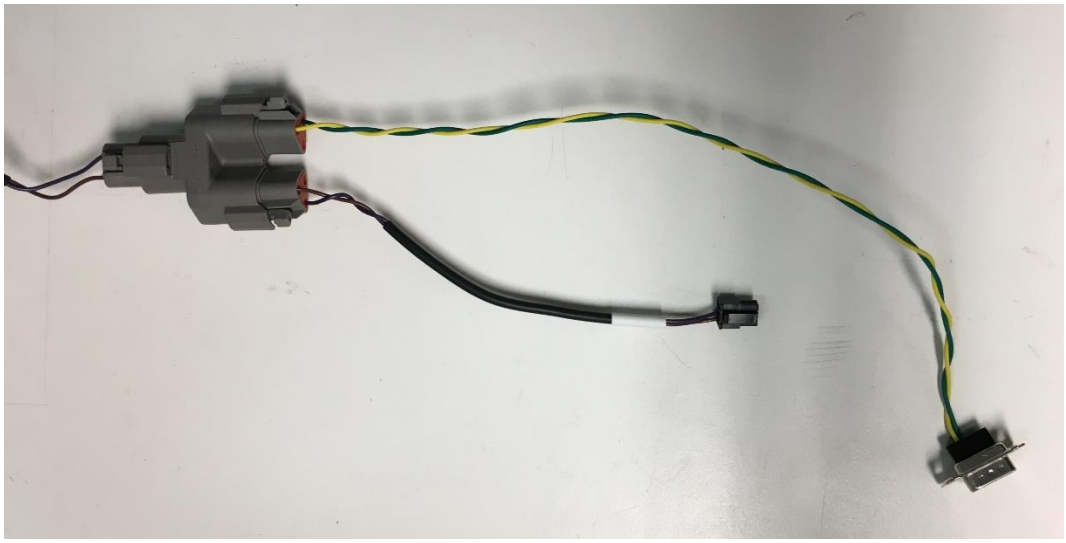

Figure 10: Mobileye 6 Data Logging CAN Connection Split and Outfitted with DB9

The final setup can be seen in Figure 11 Figure 12. The computer generates simulations that are sent to the Samsung digital display. The computer also sends messages to the CAN bus so that the Mobileye can observe them. Then, using the vehicle information on the CAN bus and the virtual environment it is observing, the Mobileye's computer vision generates valuable information to be returned to the user on the computer.

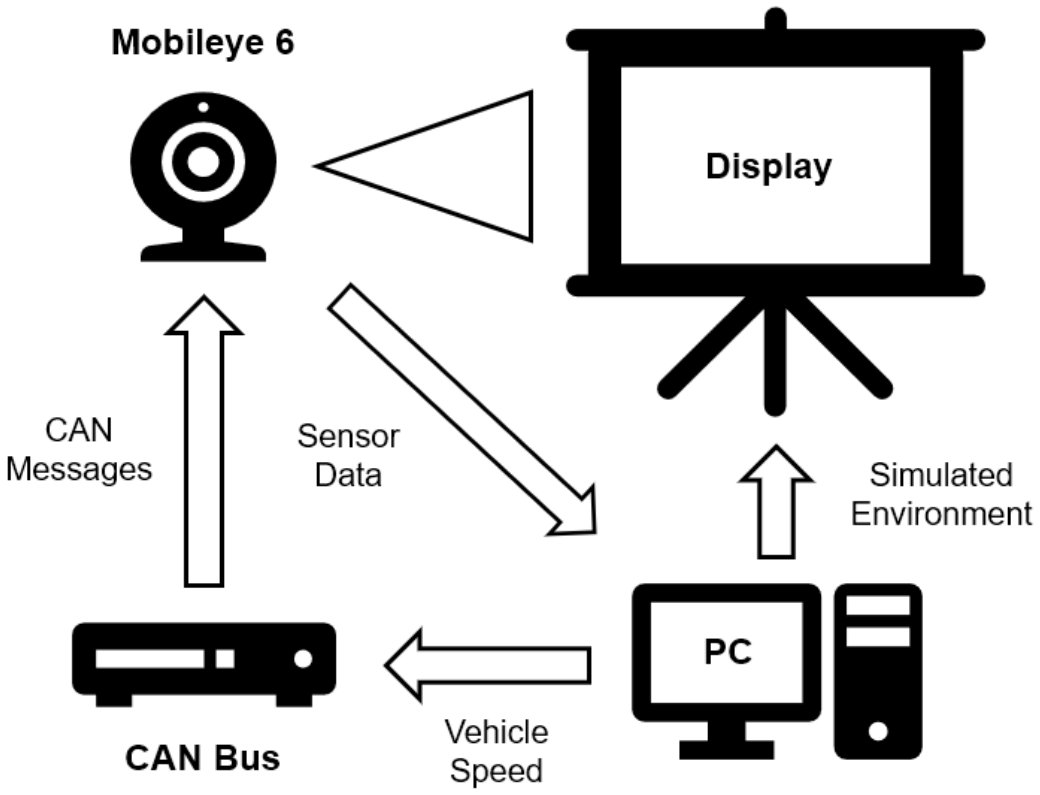

Figure 11: Diagram of CIL Setup

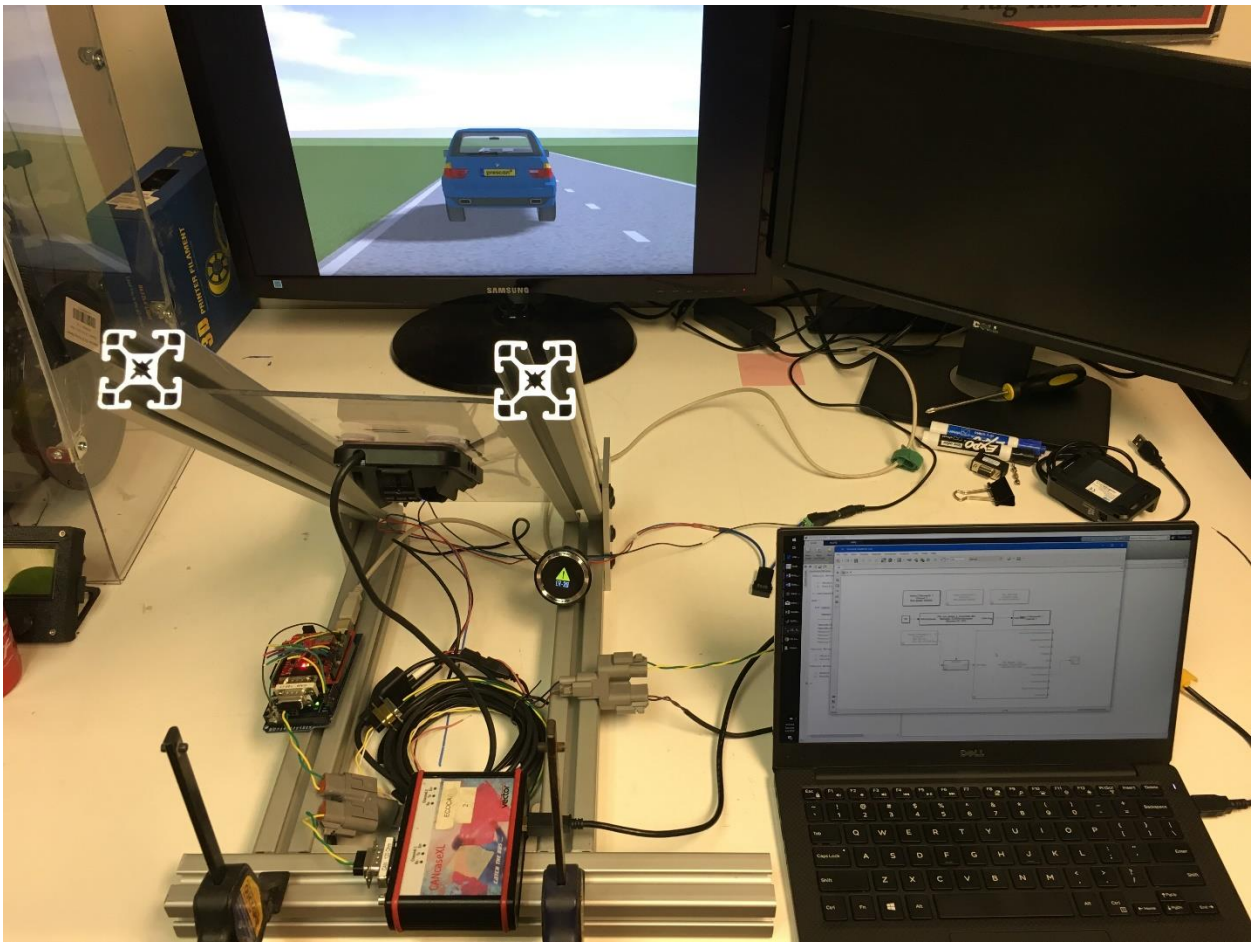

Figure 12: Final Implementation of CIL Test Bench

3 Virtual Environment

To replicate the view of a typical windshield mounted camera for the CIL, a virtual environment simulator needed to be chosen. The top priority was that the simulator would be of high enough fidelity to adequately replicate the real-world. If the camera sensor did not identify the displayed obstacles correctly, the simulator was inadequate. In conjunction, the simulator needed to be able to do so quickly and easily, allowing for rapid prototyping of control algorithms by building new test cases. Additionally, for ease of testing and application of future work, it would be ideal for the software to inherently be designed for

sensor testing. This capability should allow for a high degree of control over the sensor modules and native interaction with control algorithms. These features are standard in PreScan, making it an excellent choice for this application.

3.1 PreScan

According to Tass, the developer of PreScan, “PreScan is a physics-based simulation platform that is used in the automotive industry for development of Advanced Driver Assistance Systems (ADAS) that are based on sensor technologies” (Tass, n.d.). To accomplish this, PreScan uses a simple graphical user interface that utilizes drag-and-drop components of a simulation like other vehicles, obstacles, and even road signs. Typically, sensors would be modeled and validated in PreScan by importing your control systems to the associated Simulink model and running custom or standardized test cases. However, for the camera-in-the-loop test bench, the immediate requirements are environment visualization, sensor control, and ease of use. The tools required to do so are shown in Figure 13.

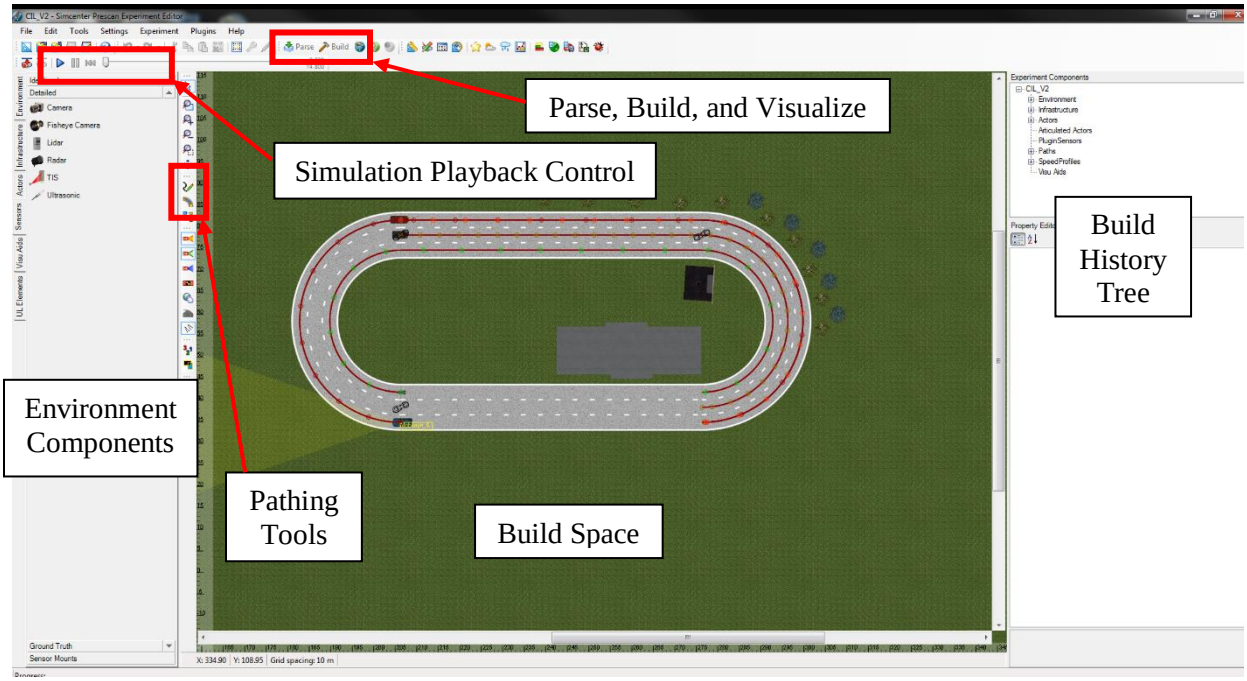

Figure 13: PreScan Graphical User Interface

Figure 14: PreScan Built Environment Components, Environment (Left), Actors (Center), Sensors (Right)

In PreScan, most of the components essential to constructing and simulating a virtual environment can be composed using the drag and drop components. For instance, to create the roadway and for the simulation, grass can be dragged into the building area. Additionally, a vehicle to serve as the ego vehicle and target vehicles can also be dragged in by changing to the tab for those respective elements. Lastly and most importantly, the sensors that are being tested in PreScan can be added via the sensor tab, which is where they can be dragged from and dropped onto the vehicle they are attached to [Figure 14]. Each sensor can be customized and modified to more accurately model its respective physical sensor. Outside of the drag-and-drop environment, the path that the vehicles follow can be manually or automatically generated by interacting with the intersections of the roadways constructed by the user [Figure 13].

After the desired environment has been constructed, the model must be prepared for visualization, the process that allows the user to observe the model carrying out the commands that they created. To do so, the parse and build buttons must be selected [Figure 13]. If errors or warning are identified, they should be addressed and the build process redone as in Figure 15. After successfully building the environment, the visualization is evoked by selecting the visualize button [Figure 14]. Upon doing so, the PreScan visualizer window opens and the various views can be used to watch the model run from different perspectives [Figure 16].

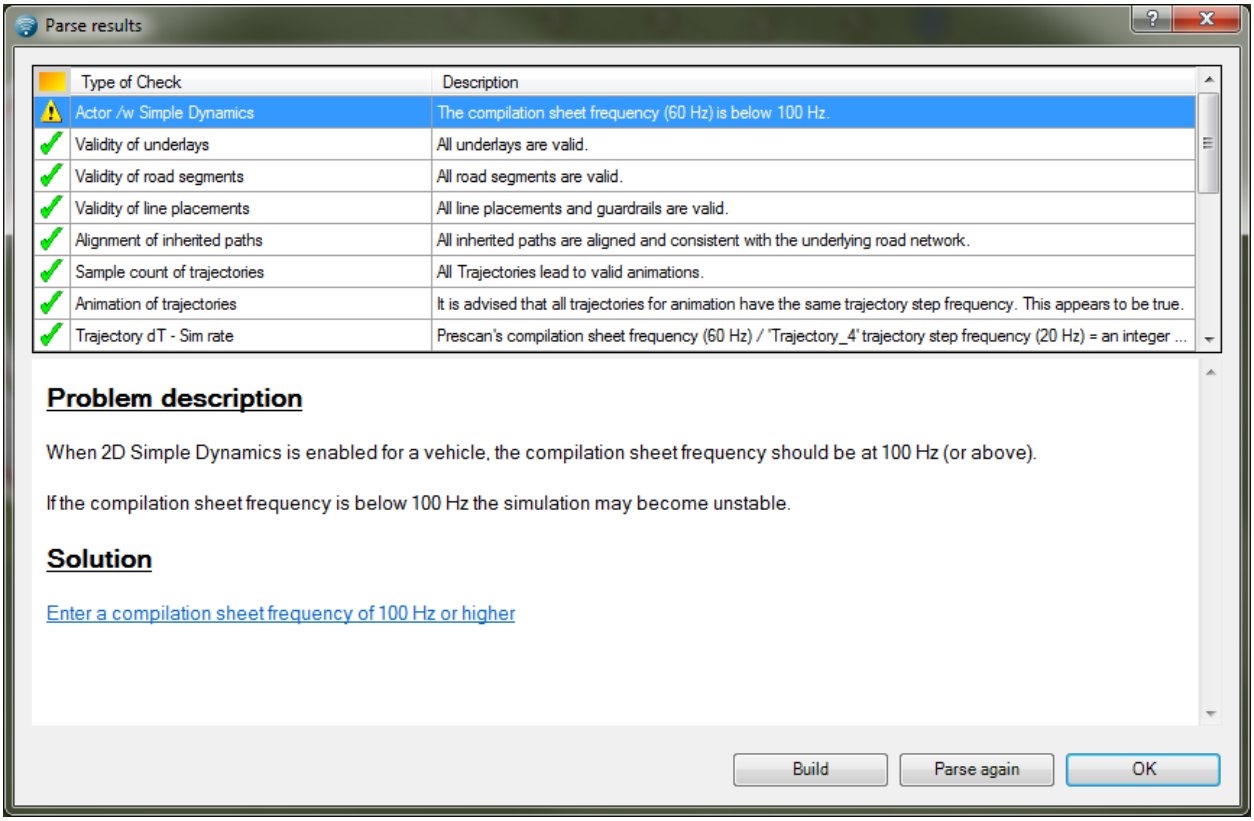

Figure 15: PreScan Parsing the Virtual Environment to Check for Problems and Preparing to Build

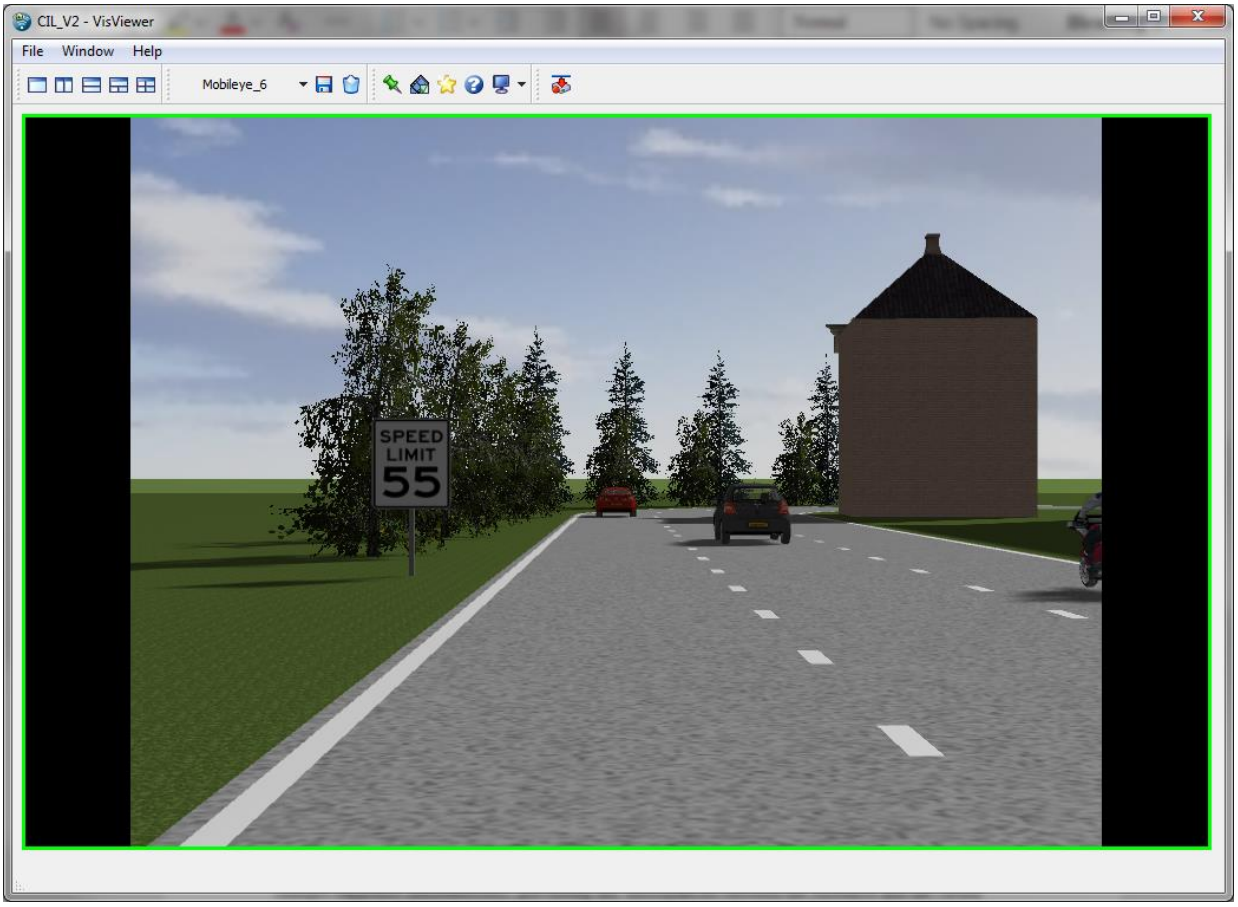

Figure 16: PreScan Visualizer from Mobileye 6 Camera Perspective

3.2 Vehicle Characterization

In PreScan, the user can modify the sensors and ego vehicle characteristics to force them to fit desired test case. For the CIL, the only sensor being utilized in PreScan is a camera sensor which is modeled using the camera sensor object. The specifications can be modified in the sensor settings to match that of the Mobileye 6 camera sensor [Figure 18]. The mounting location of the camera sensor is also altered to match what would be the actual mounting location in the physical vehicle, a 2019 Chevy Blazer for our test case (Chevrolet, 2019). These modifications can be observed in Figure 17. Additionally, for the CIL to function properly, the speed of the vehicle in PreScan must match the speed

being reported to the Mobileye unit. To set the vehicle speed in PreScan, the path that the actor follows must be modified [Figure 19]. In the path options, the speed plot can be modified. To set a constant speed, the "constant speed/accel" option was selected and the initial speed was set to match the final speed, which was the speed desired by the user [Figure 20]. Other variable speeds can be set by utilizing the other speed options in this window.

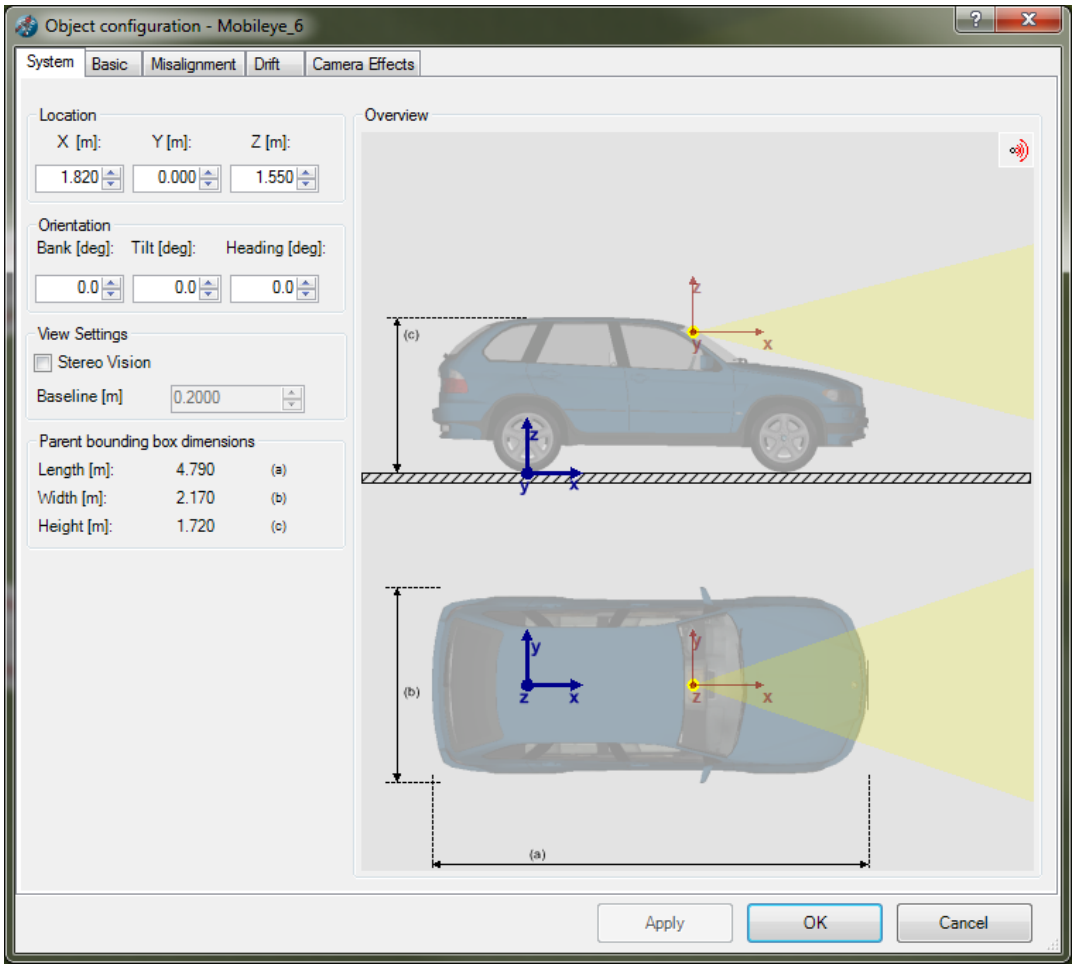

Figure 17: PreScan Camera Sensor Mounting Location Modification

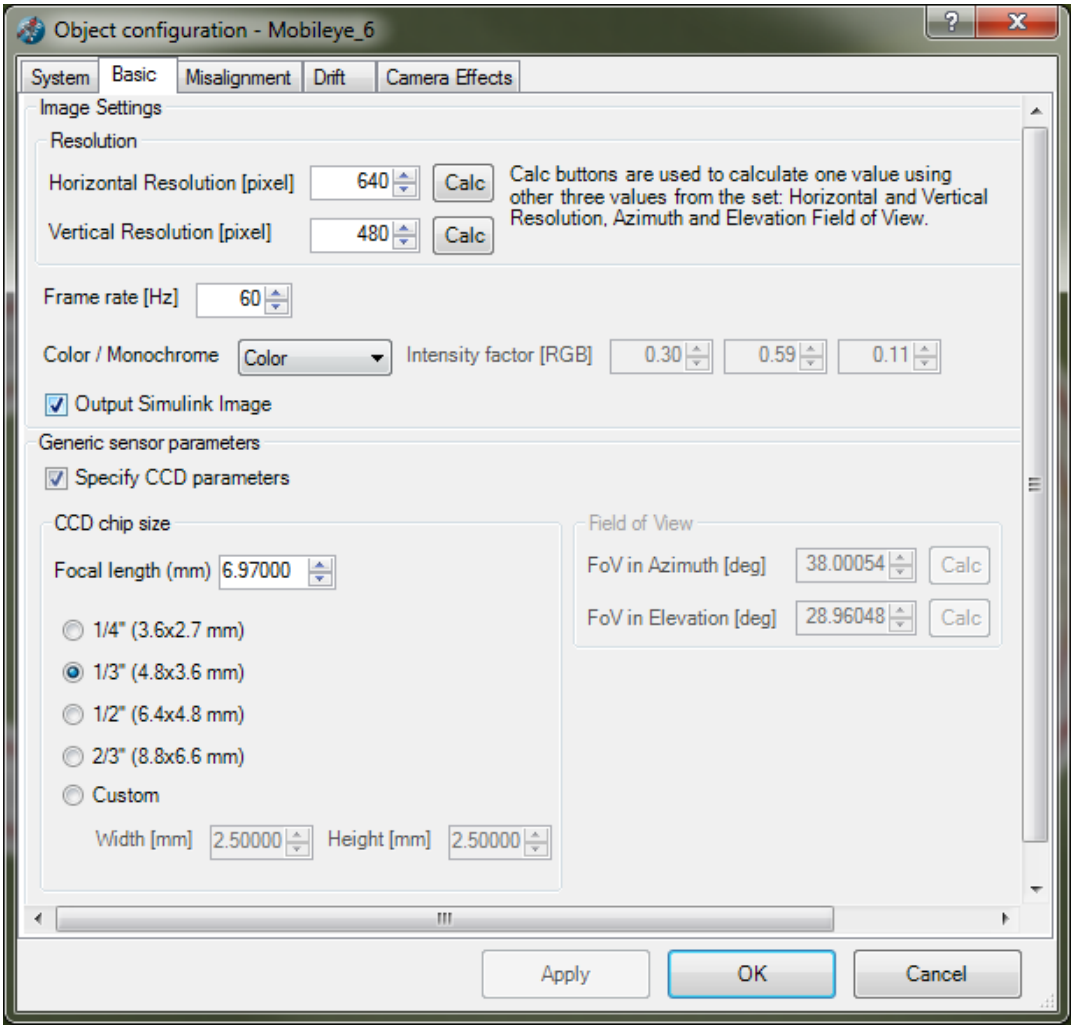

Figure 18: PreScan Camera Sensor Image Settings

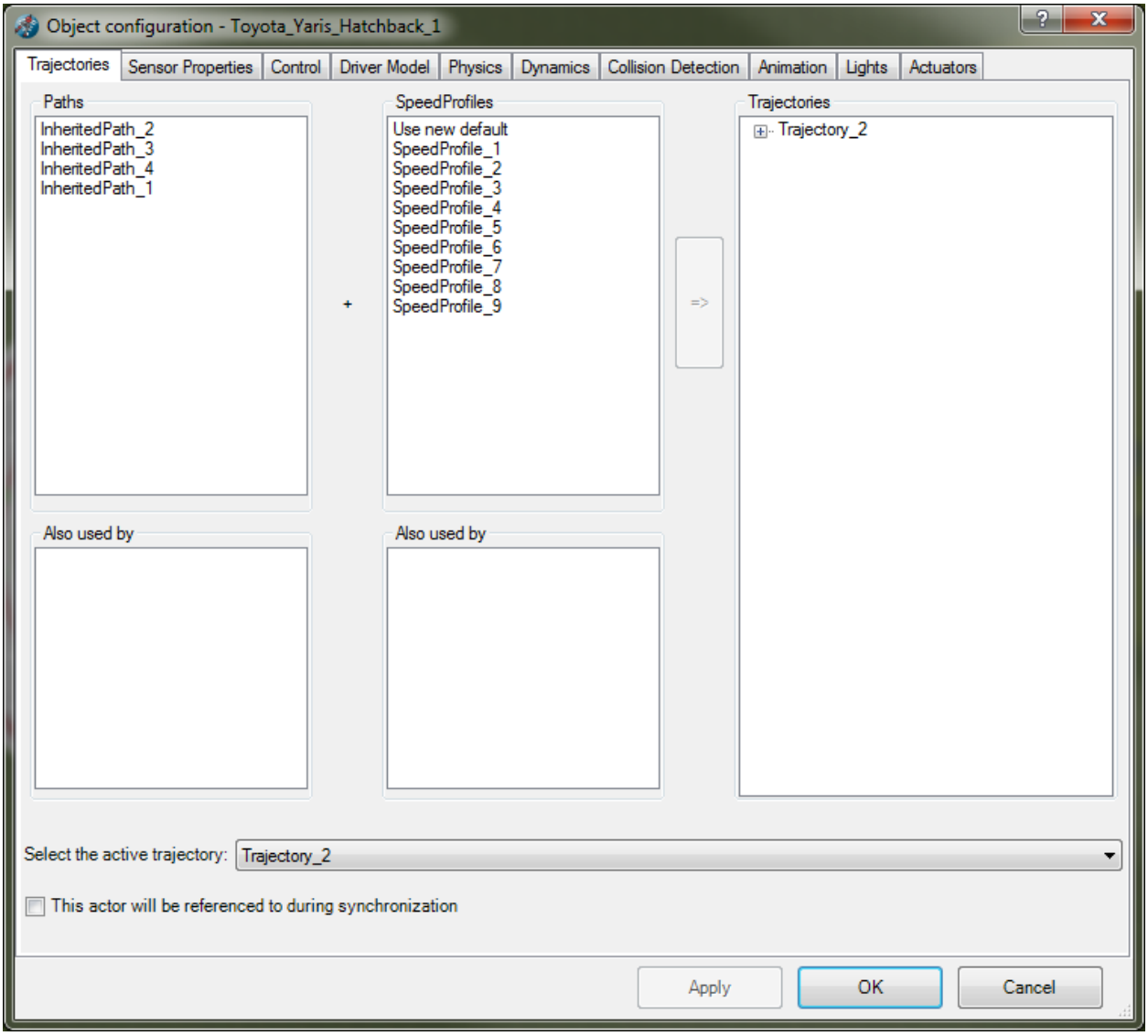

Figure 19: PreScan Actor Pathing Modification

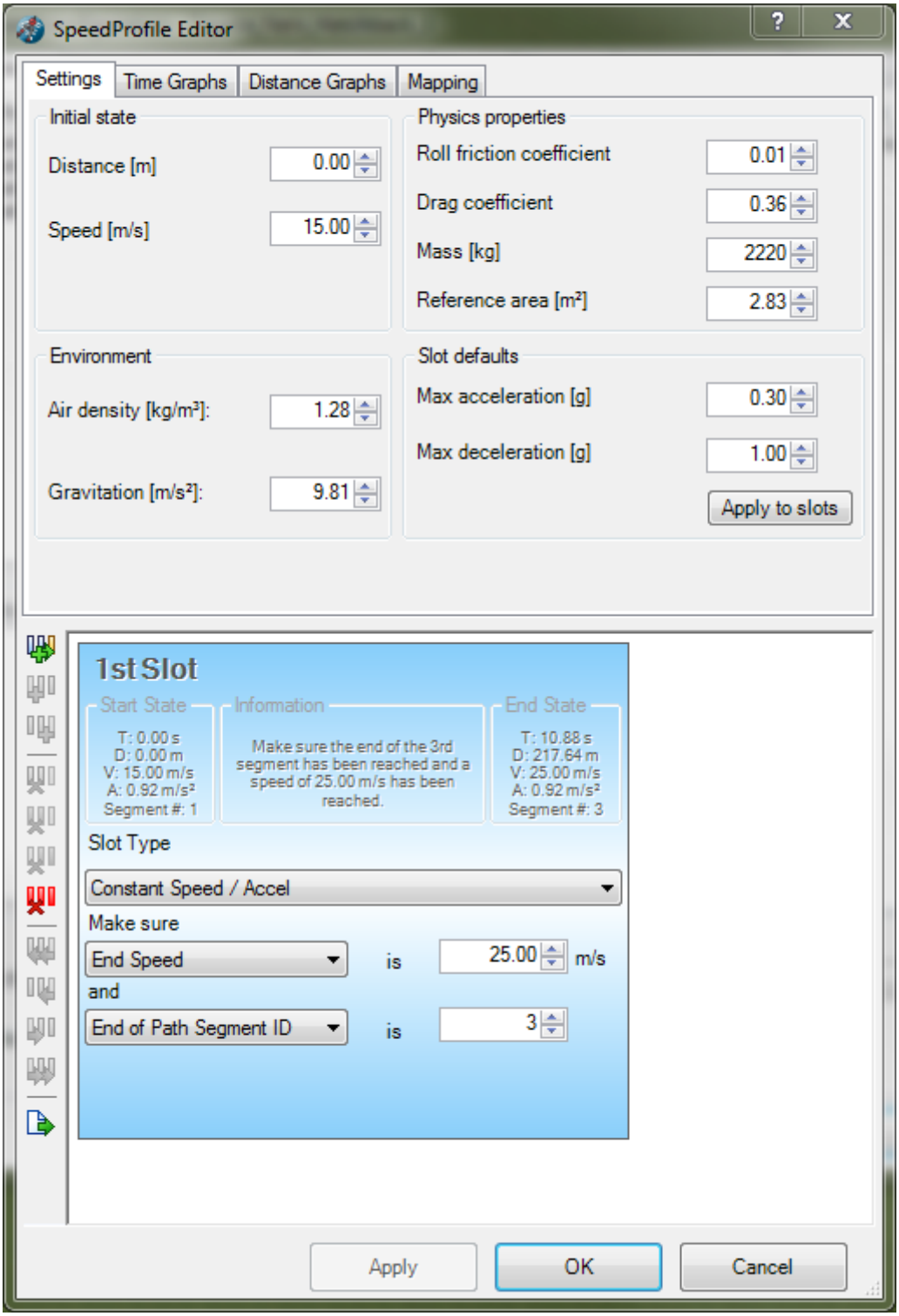

Figure 20: PreScan Actor Speed Modification

For simplicity, only constant speeds were sent to the Mobileye via the CAN bus. If variable speeds were desired, the CAN bus and the PreScan model would require synchronization to ensure that their changes happened simultaneously, preventing any discrepancies between the Mobileye and the virtual environment. This could be accomplished with identical speed curves being sent over CAN as are being set in PreScan. However, with this case, timing might become an issue. Alternatively, the CAN messaging could be integrated with PreScan, ensuring that the speed reported to the Mobileye always matches that of the virtual ego vehicle. This option is elaborated upon in the Vehicle Emulation and CAN Messaging chapter as well as in the Future Work section.

4 Calibration

The calibration process is vital to the implementation of a camera-in-the-loop test bench. It is what allows for the Mobileye unit to accurately use its computer vision to identify and locate objects in its field of view by establishing its location in space and the relative size of the objects it sees. However, the process recommended by Mobileye, the "Physical Calibration", is not viable if the user needs to test many vehicle configurations as it would require multiple vehicles, introduce error, and be time consuming. A novel solution for this was proposed to suit the rapid prototyping enabled by the CIL by way of "Digital Calibration".

4.1 Physical Calibration

In one of the recommended calibration processes used by the Mobileye 6, a Mobileye needed to be installed to a physical vehicle. This process includes mounting to a windshield, supplying power, and adding the Mobileye to the CAN bus [Figure 3]. Then, a calibration tool called a TAC board had to printed, mounted, and set up in front of the vehicle such that it could be viewed by the Mobileye.

However, this process is time consuming considering the length of the physical installation process and the time required to make the TAC board. Additionally, because this calibration process relies on the user to take measurements of the vehicle setup and physically relocate the TAC board, potential for a small user error is created every time a component is moved (Mobileye, 2016). So, in the case that the Mobileye or any control algorithms wanted to be tested on a new vehicle, it would be both time consuming and potentially inaccurate. Several images from this process can be seen below [Figure 21 Figure 22].

Figure 21: Physical Calibration Process for Mobileye with TAC Board (Mobileye, An Intel Company, 2014)

Figure 22: Physical Calibration Process for Mobileye, Taking Measurements (Mobileye, An Intel Company, 2014)

4.2 Digital Calibration

In order to address the concerns of a physical calibration process, a more suitable and robust method was put into practice for the camera-in-the-loop. In short, the camera sensor was calibrated using the digital display and a virtual TAC board rather than any physical setup. Instead, it only had to be installed into the CIL as it would be used for testing. To accomplish calibration, the standard Mobileye TAC board calibration process was performed with the following changes. First, the characteristics of the digital test vehicle are what is input to the Mobileye for calibration rather than the measurements from a physical installation. These vehicle characteristics can be pulled from the PreScan model and the ideal case that is being tested. Additionally, the TAC board used for calibration was loaded into PreScan as a traffic sign using the pdf file that it is provided by Mobileye [Figure 23]. This allows for the TAC board's size and position to be precisely controlled in the simulation environment. During the calibration process, if the

CIL mount is not perfectly positioned and the camera is not aligned exactly perfectly, slight accommodations can be made using Mobileye's calibration tool. It allows for the user to remove sections of the camera view if they are obstructed by the vehicle hood. However, in the case of CIL, this feature can be utilized ignore interference from the monitor [Figure 24].

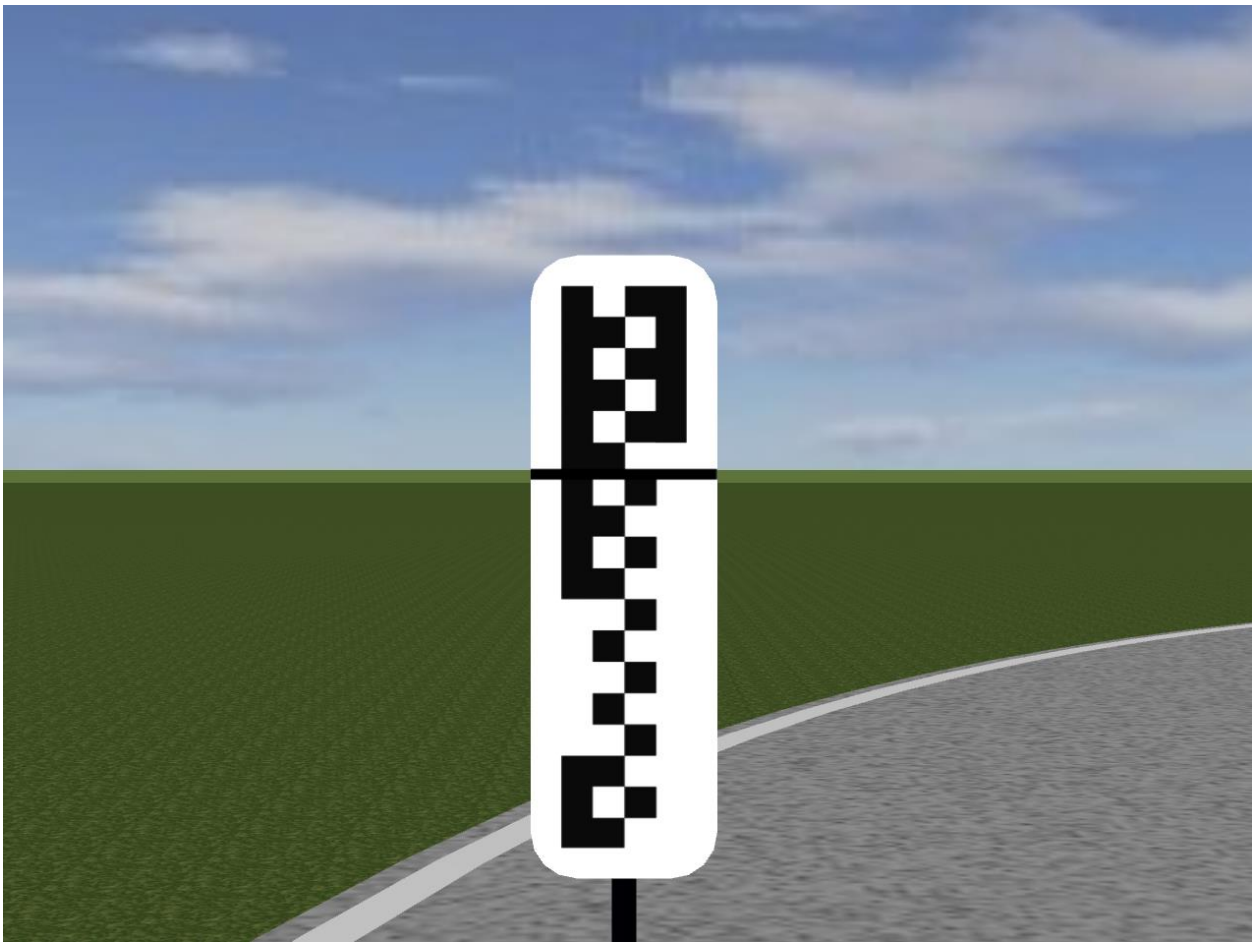

Figure 23: TAC Board for Calibration Loaded into PreScan

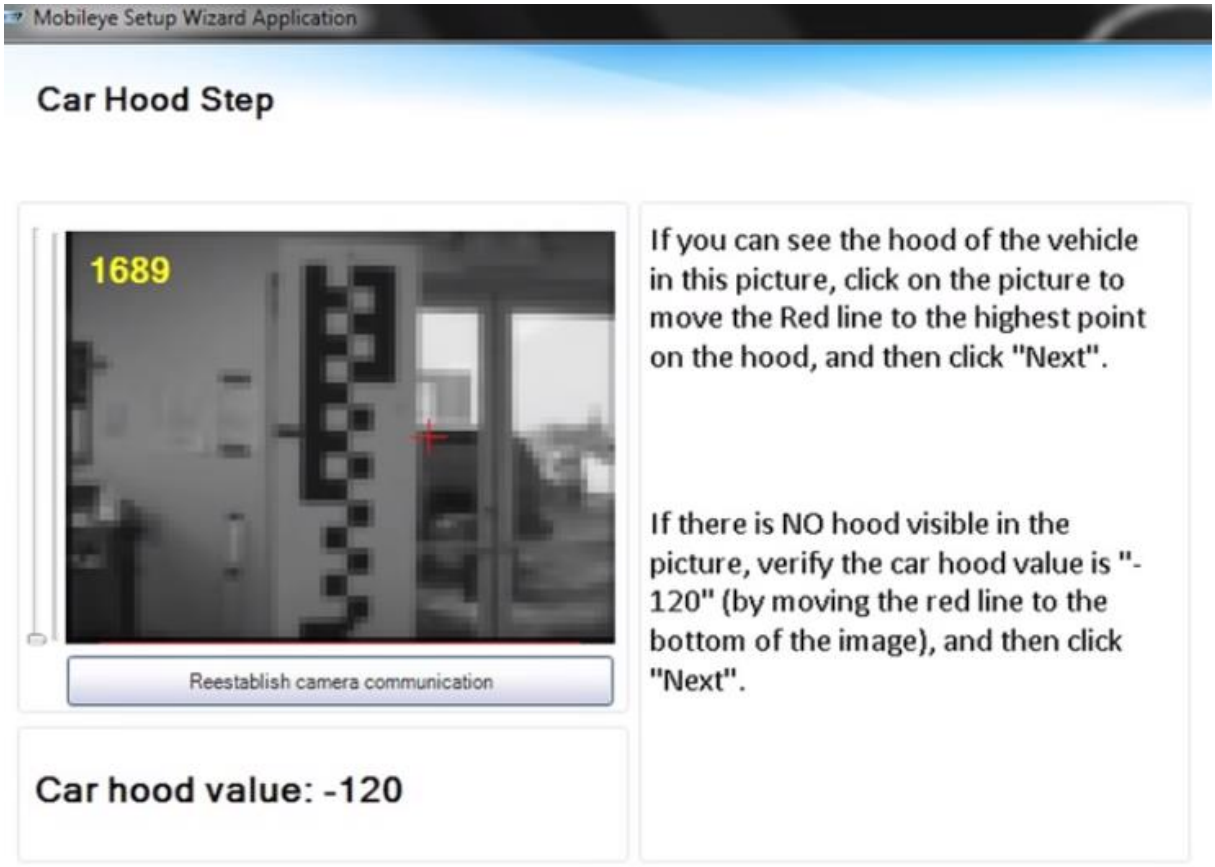

Figure 24: Mobileye Calibration Process, Car Hood Interference Setup (Mobileye, An Intel Company, 2014)

5 Vehicle Emulation and CAN Messaging

The majority of can messaging was performed in MathWorks Simulink with CAN Messaging Toolbox. A diagram of the model used to send and receive messages to the CAN bus and from the Mobileye can be seen below. Messages were sent with the CAN Pack and CAN Transmit blocks by referencing dbc files received from an open source project from Comma.ai on Github (GM Global Powertrain dbc, 2019). Those CAN messages were uploaded to the CANcaseXL from Simulink. In Simulink, the only vehicle information that must be sent to the CAN bus and Mobileye is vehicle speed, which should match the PreScan model. However, additional information such as the vehicle wiper status

must also be sent through the Simulink model for calibration. To receive data, a similar process was performed with the CAN receive block and the CAN unpack block. However, the CAN messages from the Mobileye are interpreted using a proprietary dbc file provided by Mobileye. The model that accomplishes these tasks is below [Figure 25].

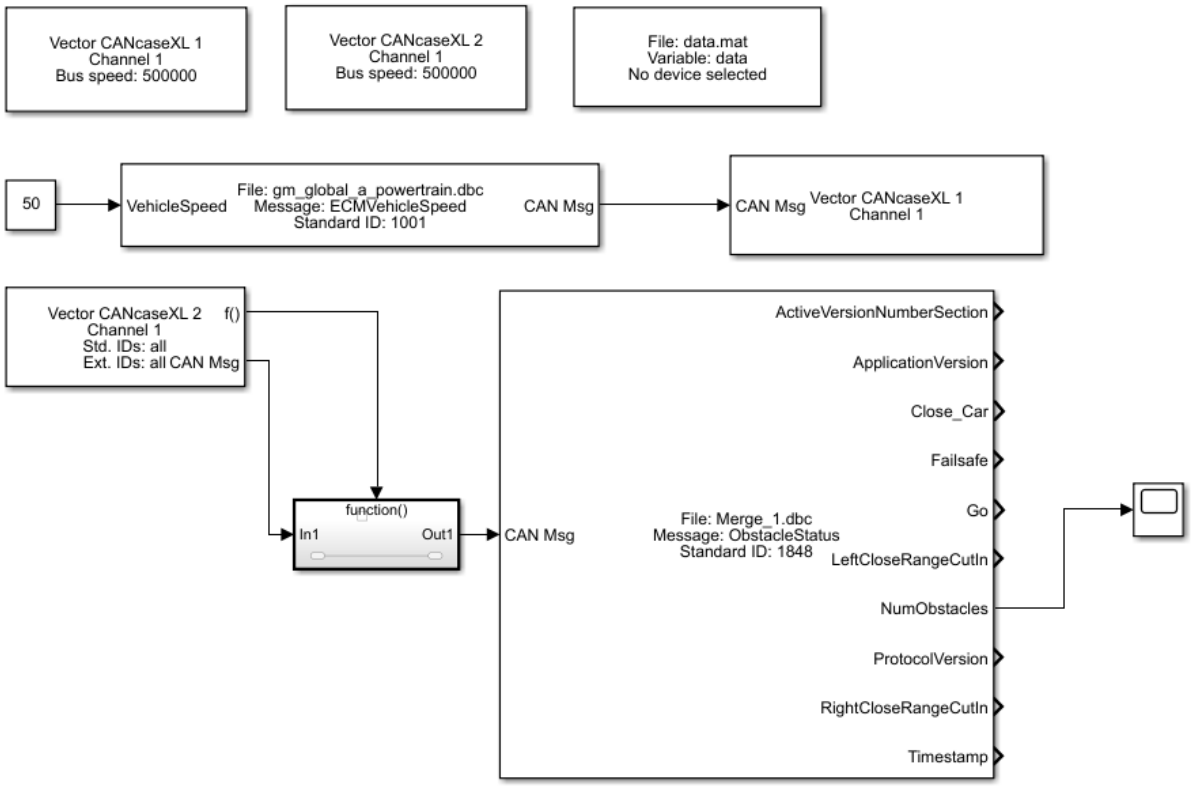

Figure 25: Simulink Model Used for CAN Messaging

6 Validation

Unfortunately, after having established the CAN messaging and data logging from the Mobileye unit, a substantial problem was observed: the only data being logged was signal data. This information informs the user of the status of the device. However, the desired information such as obstacle position, obstacle identification, and time to collision were not included in the Mobileye outputs. After further investigation with Mobileye representatives, it was revealed that two models of Mobileye units were available, developer and consumer. The unit that was purchased for this research was a consumer model, meaning that it could only output status signals. In order to log more detailed data, a developer version would need to be acquired and installed in place of the consumer unit. Otherwise, the bench seemed fully functional.

Despite this obstacle, the most vital information could still be observed from the EyeWatch unit of the Mobileye and physically recorded. Initially, this was challenging to record when playing videos on the digital display due to the high frequency of change in EyeWatch data. But, by only displaying one static image at a time, the information on the EyeWatch can only update when the user changes the displayed image. Therefore, this method of "static testing" was used as it allows for the Mobileye unit's output data to be logged reliably.

For the following tests, similar conditions were used in PreScan. First, a two-lane road with a lane width of 2 meters was placed in a grass environment. The ego vehicle was place in the center of the left lane of the road. A forward-facing camera sensor was placed at a height of 61 inches and 81.5 inches behind the hood of the vehicle. The camera was also adjusted to be level with the ground and have identical camera conditions as the Mobileye: 1/3" optical size, 38-degree horizontal field of view, and a

60 Hz refresh rate. The settings can be seen in Figure 18. This camera sensor was used as the view output in the PreScan Visualizer to be used for the CIL test bench.

6.1 Headway Distance and Forward Collision Warning

To test for forward collision detection, a PreScan environment was created in which the ego vehicle observed another vehicle, the target vehicle, that was placed in front of it in the same lane at varying distances. The target vehicle started one meter ahead of the ego vehicle and was incrementally moved forward by one meter until it was outside of a dangerous range for that speed (2.5 seconds until collision) [Figure 26]. For forward collision warnings, the EyeWatch unit outputs the time until collision with the target vehicle based on the ego vehicle's speed. These collision times were recorded at various speeds and distances [Figure 27]. Using the following formula, the distance until the collision according to Mobileye could be calculated:

$$Ego\ Vehicle\ Speed * Time\ to\ Collision = Distance\ to\ Target\ Vehicle$$

Figure 26: Forward Collision Testing, Close-range Target (Left), Middle-range Target (Center), Long-range Target (Right)

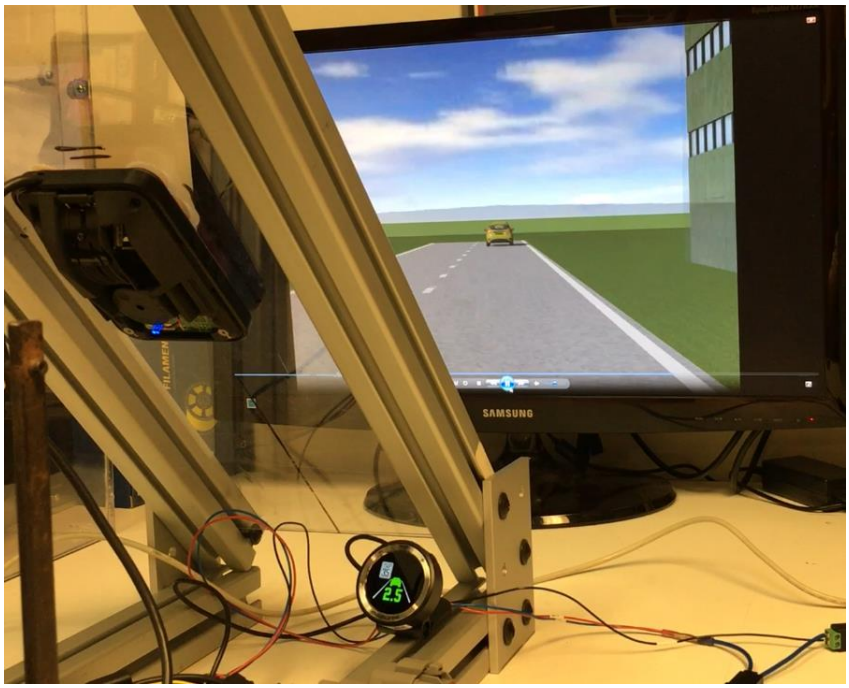

Figure 27: Mobileye FCW Detecting A Target 2.5 Seconds Away

The distance to the target vehicle at 20 mph, 30 mph, and 50mph is compared to the known distance to the target vehicle in PreScan in the following plots [Figure 28, Figure 29, and Figure 30]

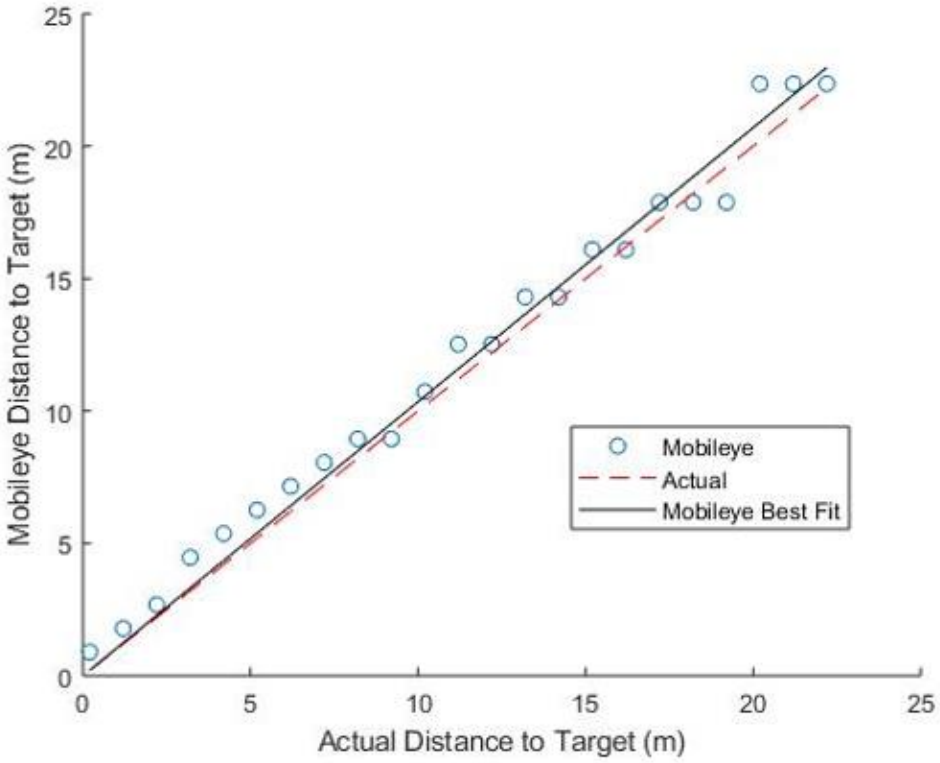

Figure 28: Forward Collision Warning Tests at 20 MPH

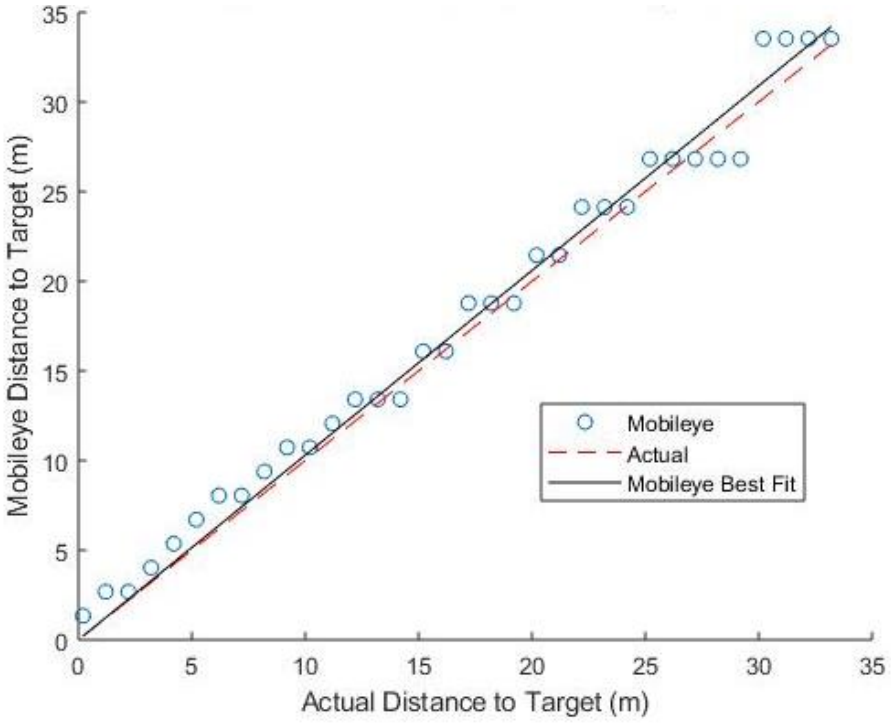

Figure 29: Forward Collision Warning Tests at 30 MPH

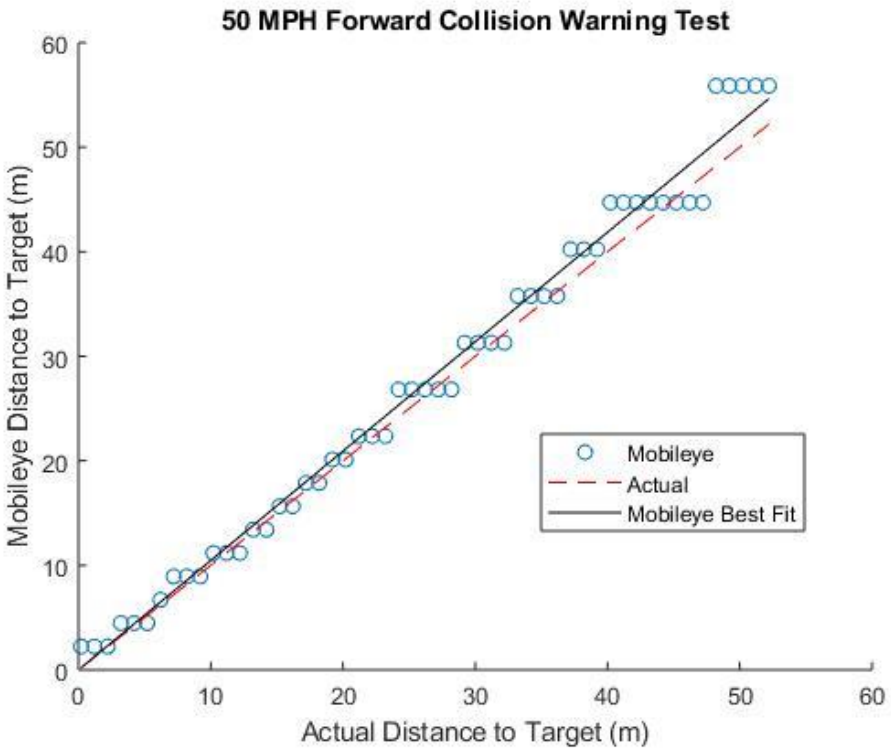

Figure 30: Forward Collision Warning Tests at 50 MPH

Upon initial inspection, it can be observed from the best fit lines that the data output by the Mobileye at all speeds generally fits the trend of the true distances. Specifically, the 30 mph test shows the closest fit at a 3.00% error between the trendline and the actual distances. However, points also can be seen grouping together at various distances according to the Mobileye. Unfortunately, this is a consequence of the low degree of precision of the EyeWatch unit. It was observed that at collision times under one second, the resolution is 0.1 seconds. From one second to two seconds, the resolution worsens to 0.2 seconds. If a vehicle is within warning range, 2.7 seconds, but it will take longer than two seconds, only a warning time of 2.5 seconds is displayed. The result of the low resolution is the introduction of large rounding errors, especially at low distances. The percent errors of these measurements follow [Figure 31, Figure 32, and Figure 33].

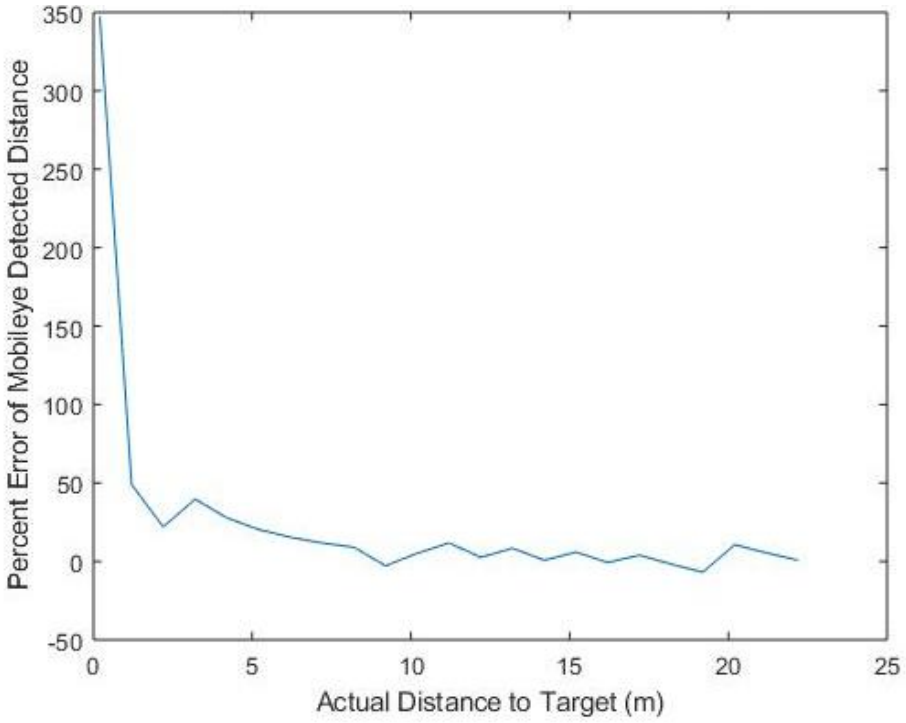

Figure 31: Forward Collision Warning Test at 20 MPH Percent Error

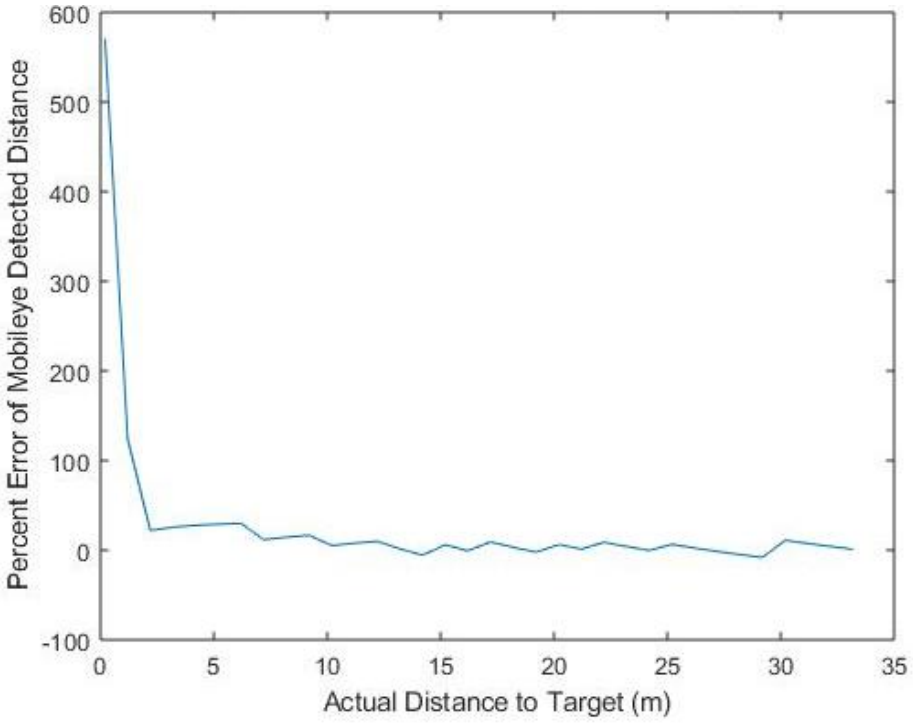

Figure 32: Forward Collision Warning Test at 30 MPH Percent Error

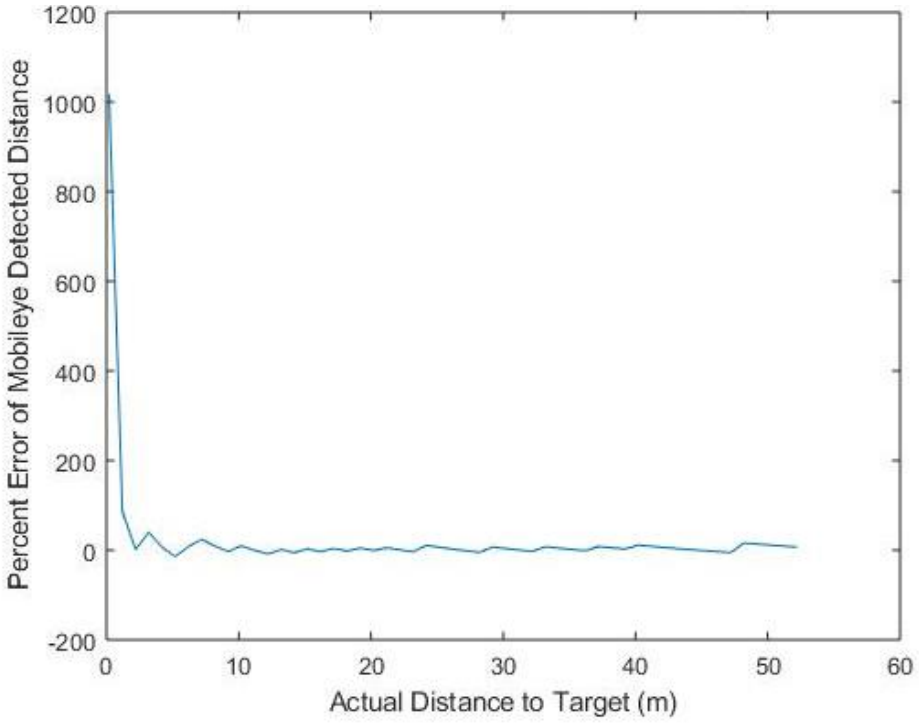

Figure 33: Forward Collision Warning Test at 50 MPH Percent Error

At low distances, the percent error of all tests was tremendous. However, as previously stated, this is a result of the EyeWatch rounding. This results in an average percent error of approximately 27.5% across the three tests. However, if the results at low distances where the rounding errors create outliers are removed, the results greatly improve [Figure 34, Figure 35, and Figure 36].

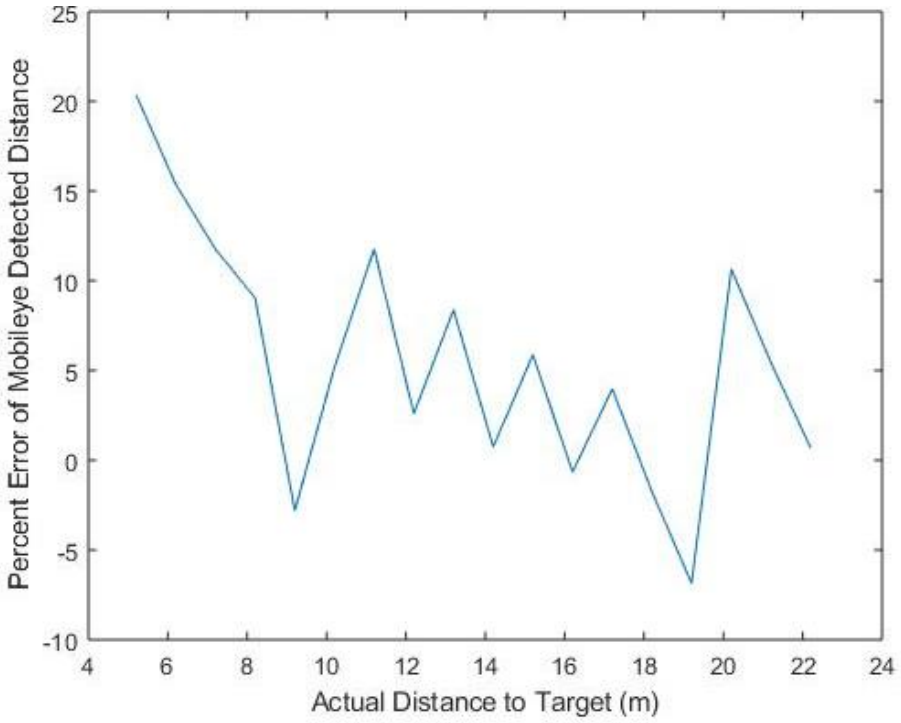

Figure 34: Forward Collision Warning Test at 20 MPH Percent Error, Outliers Removed

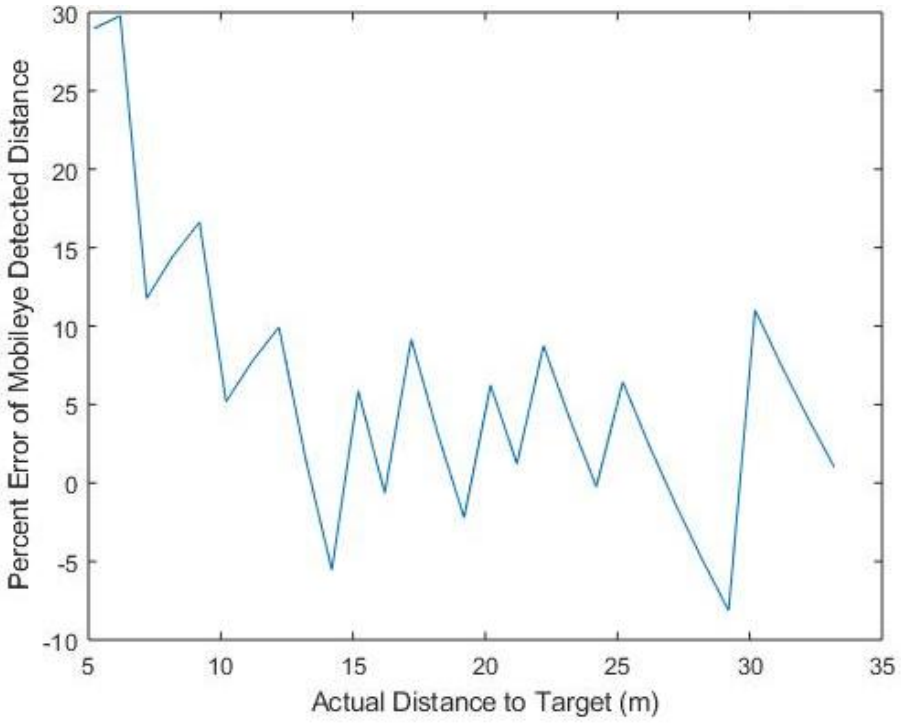

Figure 35: Forward Collision Warning Test at 30 MPH Percent Error, Outliers Removed

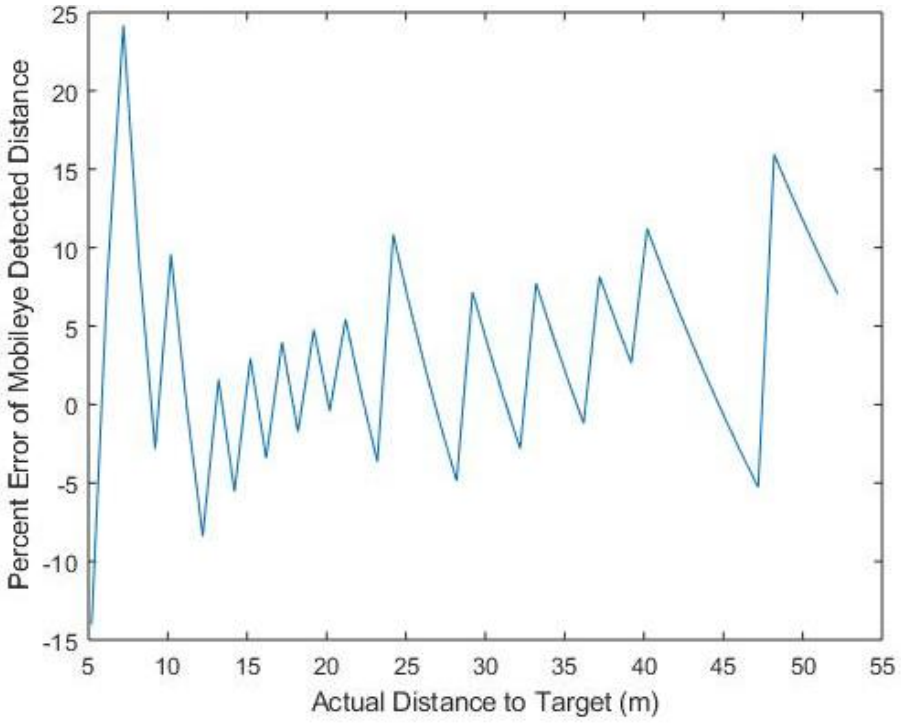

Figure 36: Forward Collision Warning Test at 50 MPH Percent Error, Outliers Removed

By removing the distances less than five meters from the measurements, the outliers, the percent error of all tests dropped. In the case of average percent error, all the tests now yielded values less than a third of the original. The percent error of the best fit lines also improved in all cases [Table 2].

Table 2: FCW Original and Adjusted Results

Speed (mph)	Original		Adjusted	
	Average Percent Error (%)	Percent Error of Best Fit Line (%)	Average Percent Error (%)	Percent Error of Best Fit Line (%)
20	26.49	3.49	6.88	3.24
30	29.10	3.00	7.59	2.92
50	26.98	4.69	5.79	4.68

6.2 Pedestrian Collision Warning

To test the Mobileye's ability to identify pedestrians and reliability in doing so, the target vehicle from the previous forward collision warning has been replaced by a pedestrian [Figure 37]. The pedestrian still moves at a resolution of one meter while the ego vehicle and camera sensor have also remained the same. From this test, a greater understanding of how the Mobileye identifies pedestrians can be learned.

Figure 37: Pedestrian Collision Testing, 30 meter Target (Left), 20 meter Target (Center), 10 meter Target (Right)

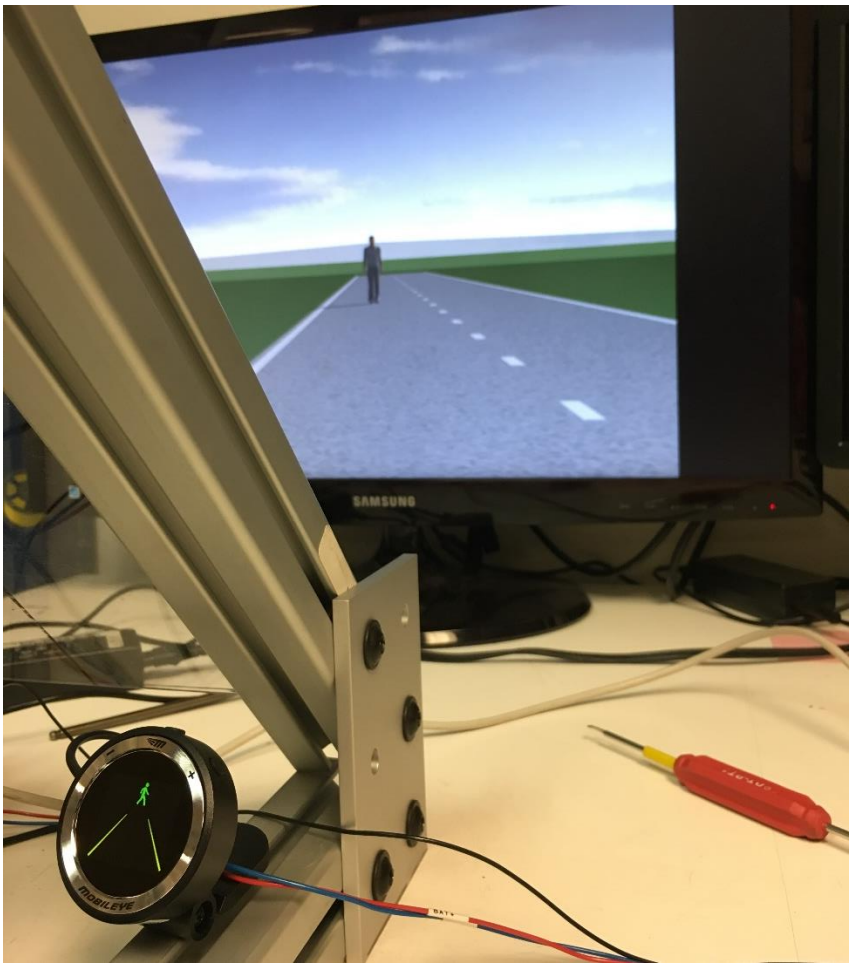

Figure 38: Mobileye Pedestrian Collision Warning Detecting a Pedestrian

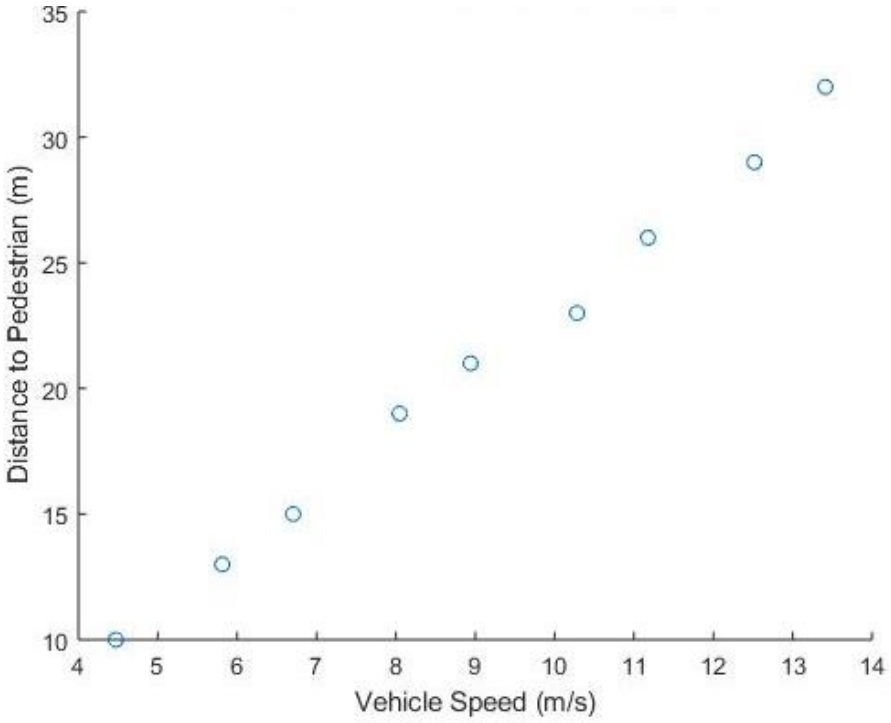

Figure 39: Pedestrian Collision Warning Pedestrian Detection Distance at Various Vehicle Speeds

In Figure 39, the pedestrian detection distance follows a linear trend in relation to the vehicle speed. At distances further than the detection point, the pedestrian could be identified. At distances less than it, it could not. A line of best fit indicates that the pedestrian is initially detectable when he or she is more than 2.32 seconds away. Although, the pedestrian can be tracked inside that range if it is initially detected outside of the 2.32 seconds range. This indicates that the CIL is capable of reliably identifying pedestrians, despite the speed. However, it also reveals that the Mobileye computer vision might not be capable or designed to object classify pedestrians inside the collision time range. Additionally, due to the lack of collision times for pedestrian collision warning on the EyeWatch, any systematic errors in distance interpretation could not be identified.

6.3 Lane Departure Warning

Lastly, to check the Camera-in-the-loop's ability to detect lanes, all obstacles in front of the ego vehicle in PreScan were removed. Additionally, as is required by the Mobileye for Lane Departure Warning functionality, the vehicle speed was set to greater than 40 mph. For my testing, the vehicle speed was set to 50 mph in the CAN messaging on Simulink. Then, the ego vehicle was shifted left at increments of 0.05 meters until it detected that it had crossed the left lane line [Figure 41]. The location at which the departure was detected was recorded. This process was repeated for the right lane departure by shifting the vehicle right [Figure 40]. After repeating the test for both directions a total of three times, identical results had been achieved.

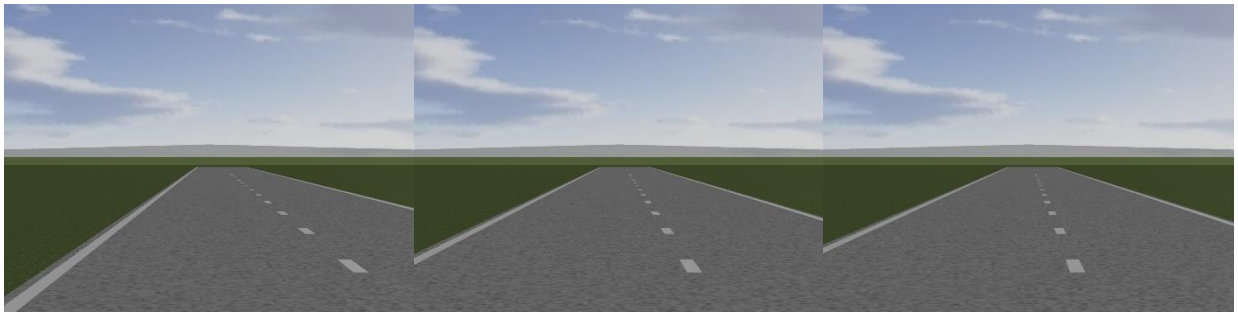

Figure 40: Lane Departure Warning Test at Center of The Lane (Left), Shifted 0.5 Meters Right (Center), and Departing The Lane (Right)

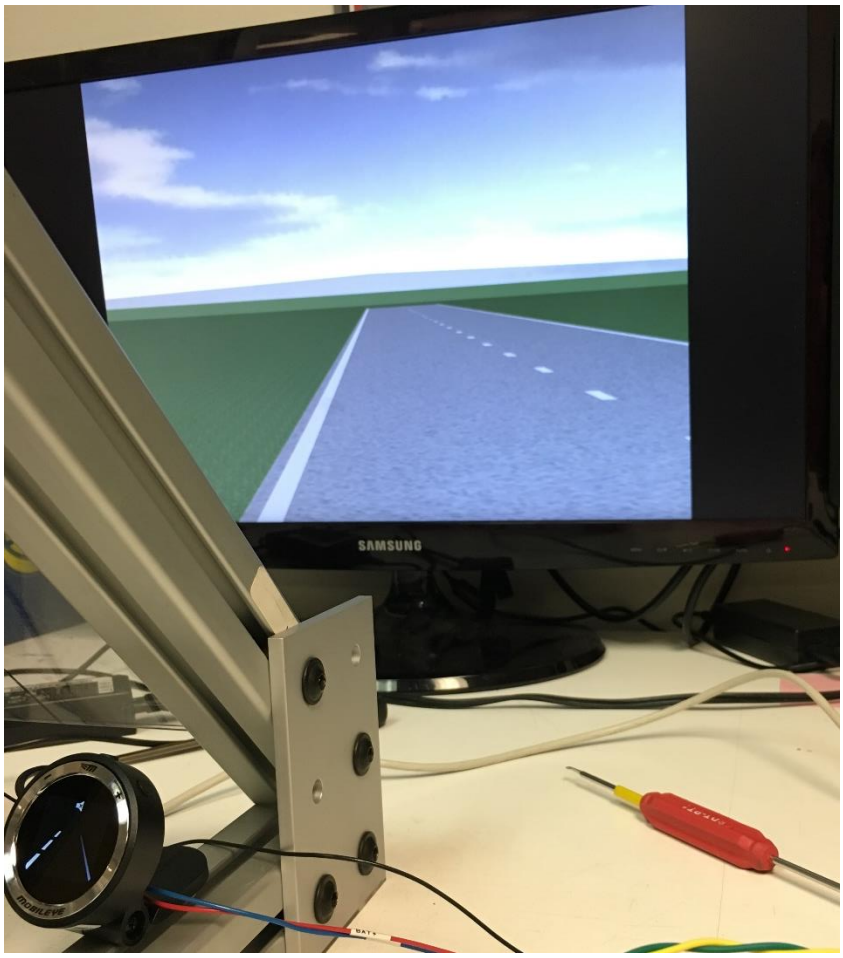

Figure 41: Lane Departure Warning Detecting a Left Side Lane Departure

Using the width of the vehicle, the width of the lane, and the distance it had been shifted from the center of the lane, the true distance between the ego vehicle's edge and the lane can be calculated.

$$Distance\ to\ Lane = \frac{Lane\ Width}{2} - \frac{Vehicle\ Width}{2} - Shift\ Distance$$

The visualization of the detected lane line is displayed against the actual location of the lane lines in the following figure. Due to the 0.05 meter resolution of the tests, the exact detected lane location could not be determined. Instead, it can be narrowed to somewhere between the detection point and the point that was shifted 0.05 meters less. The range in which the lane was identified is shown in Table 3.

Table 3: Lane Departure Warning Test Results

Direction	Detection Point Shift Distance (meters)	Distance to Line (meters)
Left	1.05	-0.0241
Right	1.20	-0.1741

As can be observed in figure, the left lane detection was very accurate with a percent error from the center of the its range of 0.05%. However, the right lane detection occurred approximately 0.15 meters outside of the right lane, resulting in a 7.45% error.

7 Conclusion

7.1 Contributions

First, this project has proven to be a viable means of validating Mobileye 6 camera sensors in various test cases. This was determined by the relatively high accuracy of the Mobileye when compared to the ground truth defined in the PreScan models. Theoretically, this process would also work for other camera sensors partnered with computer vision capabilities.

Beyond camera sensor validation, this CIL test bench also serves as an excellent hardware-in-the-loop setup for generating life like data from a computer vision camera system to use in control process testing. For instance, if automatic emergency brake controllers needed to be tested, a test case can be generated in minutes without ever driving a vehicle. Not only is this process fast and cheaper than

purchasing a test vehicle, but it is also safer. Otherwise dangerous test cases can be used because they are entirely virtual.

7.2 Future Work

In its current state with a consumer version Mobileye 6, the test bench is unable to retrieve all the data generated by the Mobileye sensor and computer vision. However, by simply replacing it with a develop version, which is available inside the Center for Automotive Research, this can be changed. Once the developed Mobileye is logging data, the CIL can be used a means of rapidly testing ADAS controllers thanks to the large amount of data provided by Mobileye's computer vision. To accomplish this, the current Mobileye would be removed, a new one installed, outfit it with connections, and connect another CAN logging device to the Mobileye CAN logging connection. By doing so, the rounding errors from the EyeWatch would also be removed, presumably improving the accuracy of the validation results even further.

However, not all error can be removed by replacing the Mobileye. Based on the one-sided error in the lane departure warning tests, it can be assumed that the camera sensor is not perfectly centered on the screen, not perfectly calibrated, or both. To improve this, the Mobileye unit's orientation would simply need to be slightly adjusted based off the placement measurements and then recalibrated to the screen. After rerunning the validation tests, it could be determined if the accuracy improved. Next, the steps would be repeated until a satisfactory level of accuracy was achieved. This process would be slow, but could help eliminate systematic error in the results, which becomes very important in ADAS controller validation.

Lastly, the Simulink model for the PreScan simulation could "close the loop" with the camera sensor. As previously mentioned, PreScan operates with MATLAB and Simulink. So, the CAN

messaging blocks for transmitting vehicle speed could simply me moved to the PreScan model. Then, the vehicle speed information from inside PreScan could be sent onto the CAN bus and thereby sent to the Mobileye in real-time during the simulation. Additionally, if controllers were to be integrated in to the Simulink environment for PreScan, the computer vision outputs from the Mobileye could be fed into the controllers and used to control the vehicle in the simulation, creating a feedback loop. Theoretically, other sensors like radar and lidar could also be modeled in PreScan, providing a platform to test sensor fusion algorithms. And while this system would allow for full testing of ADAS controllers, it would be extremely computationally demanding and require a deeper knowledge of PreScan's Simulink functionality.

7.3 Summary

In this project, a functional hardware-in-the-loop system was developed to test a Mobileye 6 camera sensor and to output test cases for ADAS controllers. The results indicate that the CIL has a relatively high degree of accuracy with a systematic error in the horizontal direction. When detecting other vehicles located in the same lane as the driver, less than 10 percent average error is maintained at reasonable distances. Although, the trend indicated by the tests are more accurate with a percent error of less than five percent in the best fit line. In the case of pedestrians, they are unable to be initially detected when they are spotted inside a certain time-based range. This is most likely due to the Mobileye's not being designed to or capable of confidently classifying pedestrians inside that range. Lastly, the Mobileye 6 is extremely accurate when detecting lanes to the left of the vehicle with an error under one percent. But the right hand side has a much greater 7.45% error, indicating the systematic error in the horizontal direction. Additionally, the CIL is not currently fully functional as it relies on a consumer version Mobileye camera sensor that does not allow for full data logging. However, the system has great potential for further

controller testing due to the capability of PreScan to generate high-fidelity test cases entirely virtually and in very little time.

Appendices

Nomenclature

HIL	Hardware-in-the-loop
CIL	Camera-in-the-loop
CAN	Controller Area Network
dbc	Filetype used in CAN messaging
CAN bus	Network of controllers that communicate using the CAN protocol
Mobileye	Camera sensor used in the CIL
EyeWatch	Small display provided with the Mobileye to provide information to a driver
PreScan	Software used to generate virtual environment simulations
Simulink	Software used for CAN messaging and logging

Specification Sheets

Mobileye 6 Main unit	
Physical Characteristics	
Length:	122mm
Width (without lens):	79mm
Height:	43mm
Weight:	200g
Color:	Black
Case Material:	Aluminum/plastic
Cable Length:	3m
Cable Diameter:	4.8mm
Electrical Characteristics	
Input Voltage:	12-28VDC
Input current:	12v → 220mA, 24v → 120mA
Environmental Characteristics	
Operating Temperature:	-20°c to +85°c
Storage Temperature:	-40°c to +105°c
Vision Sensor	
Vision Sensor:	Aptina MT9V024 (1/3") RCC
Array Format:	Total: 752H x 480V - Active pixels: 640H x 480V
Optical Format:	1/3"
Pixel Size:	6.0µm x 6.0µm
Dynamic Range:	>55dB linear; >100dB in HDR mode
Shutter type:	Global shutter—TrueSNAP™
Responsivity:	4.8 V/lux sec (550nm)
Angle of view:	38° (horizontal)
Focus range:	5m to infinity
AGC:	Automatic Gain Control of the image sensor for high dynamic range
Audio Buzzer	
SPL Minimum	**86dB @ 10cm**
EyeQ2® Vision Processor	
332 MHz clock rate running seven parallel processes	
Two MIPS24KF 32bit CPUs	
Eight 64bit Vision Computing Engines (VCE)	
Eight channels DMA	
64bit width 512KB on-chip SRAM	
Bluetooth Module	
CSR BC5 Class II v2.1	
Profile Support: Serial Port Profile (SPP)	
Pairing Code: 1234	
Mobile Platform Supported: Android, IOS* *Only Mobileye 6 models with Blue L.E.D	

Figure 42: Mobileye Camera Sensor Specifications (Mobileye, 2016)

MT9V024/D

1/3-Inch Wide VGA CMOS Digital Image Sensor

Description

The MT9V024 is a 1/3−inch wide−VGA format CMOS active−pixel digital image sensor with global shutter and high dynamic range (HDR) operation. The sensor has specifically been designed to support the demanding interior and exterior automotive imaging needs, which makes this part ideal for a wide variety of imaging applications in real−world environments.

Table 1. KEY PERFORMANCE PARAMETERS

Parameter	Value
Optical Format	1/3-inch
Active Imager Size	4.51 mm (H) x 2.88 mm (V) 5.35 mm Diagonal
Active Pixels	752 H x 480 V
Pixel Size	6.0 µm x 6.0 µm
Color Filter Array	Monochrome or Color RGB Bayer or RCCC Pattern
Shutter Type	Global Shutter
Maximum Data Rate Master Clock	27 Mp/s 27 MHz
Full Resolution	752 x 480
Frame Rate	60 fps (at Full Resolution)
ADC Resolution	10−bit Column−Parallel
Responsivity	4.8 V/lux−sec (550 nm)
Dynamic Range	> 55 dB Linear; > 100 dB in HDR Mode
Supply Voltage	3.3 V ± 0.3 V (All Supplies)
Power Consumption	< 160 mW at Maximum Data Rate (LVDS Disabled); 120 µW Standby Power at 3
Operating Temperature	−40°C to + 105°C
Packaging	52−ball iBGA, Automotive−qualified; Wafer or Die

Features

- Array Format: Wide−VGA, Active 752 H x 480 V (360,960 pixels)
- Global Shutter Photodiode Pixels; Simultaneous Integration and Readout
- RGB Bayer, Monochrome, or RCCC: NIR Enhanced Performance for Use with Non−visible NIR Illumination
- Readout Modes: Progressive or Interlaced
- Shutter Efficiency: >99%
- Simple Two−Wire Serial Interface
- Real−Time Exposure Context Switching−Dual Register Set
- Register Lock Capability

ON Semiconductor®

www.onsemi.com

IBGA52
CASE 503AA

ORDERING INFORMATION

See detailed ordering and shipping information on page 2 of this data sheet.

Features (continued)

- Window Size: User Programmable to Any Smaller Format (QVGA, CIF, QCIF). Data Rate Can Be Maintained Independent of Window Size
- Binning: 2 x 2 and 4 x 4 of The Full Resolution
- ADC: On−Chip, 10−bit Column−Parallel (Option to Operate in 12−bit to 10−bit Companding Mode)
- Automatic Controls: Auto Exposure Control (AEC) and Auto Gain Control (AGC); Variable Regional and Variable Weight AEC/AGC
- Support for Four Unique Serial Control Register IDs to Control Multiple Imagers on the Same Bus
- Data Output Formats:
  - Single Sensor Mode: 10−bit Parallel/Stand−Alone 8−bit or 10−bit Serial LVDS
  - Stereo Sensor Mode: Interspersed 8−bit Serial LVDS
- High Dynamic Range (HDR) Mode

Figure 43: Mobileye Image Sensor Specifications (Semiconductor Components Industries, LLC., November)

MATLAB Code

```
%Timothy Kirby
%Undergraduate Thesis Sp 19
%Ohio State University

clear; clc; close all;

%% Forward Collision Warning

D0 = 0.2;
precision = 1;

%% 20 MPH Forward Collision Warning

Speed_mph = 20;
Speed = Speed_mph*0.44704;

%0.1 0.2 0.3 0.5 0.6

t_Mobileye = [0.1 0.2 0.3 0.5 0.6 0.7 0.8 0.9 ...
            1 1 1.2 1.4 1.4 1.6 1.6 1.8 1.8 ...
            2 2 2 2.5 2.5 2.5];
D_Truth_20 = [D0:precision:D0+length(t_Mobileye)-1];
D_Mobileye_20 = t_Mobileye.*Speed;

m_20 = D_Truth_20'\D_Mobileye_20';
Fit_20 = D_Truth_20*m_20;

R_20 = fitlm(D_Truth_20,D_Mobileye_20)

Fit_Error_20 = (m_20-1)*100

figure(1)
hold on
plot(D_Truth_20,D_Mobileye_20,'o')
plot(D_Truth_20,D_Truth_20,'r--')
plot(D_Truth_20,Fit_20,'k')
hold off
title('20 MPH Forward Collision Warning Test')
xlabel('Actual Distance to Target (m)')
ylabel('Mobileye Distance to Target (m)')
legend('Mobileye','Actual','Mobileye Best Fit','Location','best')

Error_20 = D_Mobileye_20-D_Truth_20;
Pct_Error_20 = Error_20./D_Truth_20*100;

figure(2)
plot(D_Truth_20,Pct_Error_20, 'o')
title('20 MPH Percent Error vs Distance to Target')
```

```
xlabel('Actual Distance to Target (m)')
ylabel('Percent Error of Mobileye Detected Distance')

Average_Absolute_Pct_Error_20 = mean(abs(Pct_Error_20))

%% 30 MPH Forward Collision Warning

Speed_mph = 30;
Speed = Speed_mph*0.44704;

%0.1 0.2 0.2 0.3 0.4

t_Mobileye = [0.1 0.2 0.2 0.3 0.4 0.5 0.6 0.6 0.7 0.8 0.8 0.9 ...
            1 1 1 1.2 1.2 1.4 1.4 1.4 1.6 1.6 1.8 1.8 1.8 ...
            2 2 2 2 2 2.5 2.5 2.5 2.5];
D_Truth_30 = [D0:precision:D0+length(t_Mobileye)-1];
D_Mobileye_30 = t_Mobileye.*Speed;

m_30 = D_Truth_30'\D_Mobileye_30';
Fit_30 = D_Truth_30*m_30;

R_30 = fitlm(D_Truth_30,D_Mobileye_30)

Fit_Error_30 = (m_30-1)*100

figure(3)
hold on
plot(D_Truth_30,D_Mobileye_30,'o')
plot(D_Truth_30,D_Truth_30,'r--')
plot(D_Truth_30,Fit_30,'k')
hold off
title('30 MPH Forward Collision Warning Test')
xlabel('Actual Distance to Target (m)')
ylabel('Mobileye Distance to Target (m)')
legend('Mobileye','Actual','Mobileye Best Fit','Location','best')

Error_30 = D_Mobileye_30-D_Truth_30;
Pct_Error_30 = Error_30./D_Truth_30*100;

figure(4)
plot(D_Truth_30,Pct_Error_30, 'o')
title('30 MPH Percent Error vs Distance to Target')
xlabel('Actual Distance to Target (m)')
ylabel('Percent Error of Mobileye Detected Distance')

Average_Absolute_Pct_Error_30 = mean(abs(Pct_Error_30))

%% 50 MPH Forward Collision Warning

Speed_mph = 50;
Speed = Speed_mph*0.44704;
```

```
%0.1 0.1 0.1 0.2 0.2 0.2

t_Mobileye = [0.1 0.1 0.1 0.2 0.2 0.2 0.2 0.3 0.4 0.4 0.4 0.5 0.5 0.5 ...
              0.6 0.6 0.7 0.7 0.8 0.8 0.9 0.9 1 1 1 ...
              1.2 1.2 1.2 1.2 1.2 1.4 1.4 1.4 1.4 ...
              1.6 1.6 1.6 1.6 1.8 1.8 1.8 ...
              2 2 2 2 2 2 2 2.5 2.5 2.5 2.5 2.5 2.5];
D_Truth_50 = [D0:precision:D0+length(t_Mobileye)-1];
D_Mobileye_50 = t_Mobileye.*Speed;

m_50 = D_Truth_50'\D_Mobileye_50';
Fit_50 = D_Truth_50*m_50;

R_50 = fitlm(D_Truth_50,D_Mobileye_50)

Fit_Error_50 = (m_50-1)*100

figure(5)
hold on
plot(D_Truth_50,D_Mobileye_50,'o')
plot(D_Truth_50,D_Truth_50,'r--')
plot(D_Truth_50,Fit_50,'k')
hold off
title('50 MPH Forward Collision Warning Test')
xlabel('Actual Distance to Target (m)')
ylabel('Mobileye Distance to Target (m)')
legend('Mobileye','Actual','Mobileye Best Fit','Location','best')

Error_50 = D_Mobileye_50-D_Truth_50;
Pct_Error_50 = Error_50./D_Truth_50*100;

figure(6)
plot(D_Truth_50,Pct_Error_50, 'o')
title('50 MPH Percent Error vs Distance to Target')
xlabel('Actual Distance to Target (m)')
ylabel('Percent Error of Mobileye Detected Distance')

Average_Absolute_Pct_Error_50 = mean(abs(Pct_Error_50))

%% Pedestrian Collision Warning

Speed_mph = [10 13 15 18 20 23 25 28 30];
Speed = Speed_mph.*0.44704;

Distance = [10 13 15 19 21 23 26 29 32];

m = Speed'\Distance'

figure(7)
hold on
plot(Speed,Distance,'o')
```

```
%plot(Speed,Speed*m,'k--')
hold off
title('Pedestrian Detection Distance vs Speed')
xlabel('Vehicle Speed (m/s)')
ylabel('Detected Distance to Pedestrian (m)')

%% Lane Departure Warning

Max = 2;
Left = 1.05;
Right = 1.2;

Width = 76.7;
Width = Width*2.54/100;
Side = Width/2;

Left_Detect = Max-Left-Side;
Right_Detect = Max-Right-Side;

precision = 0.05;

Left_0 = Left_Detect+precision;
Right_0 = Right_Detect+precision;

Left_Error = (Left_Detect+Left_0)/2/Max*100
Right_Error =(Right_Detect+Right_0)/2/Max*100

Lane = [0,1];

Left_True = [-Max -Max];
Left_Out = [Left_Detect-Max, Left_Detect-Max];
Left_In = [Left_0-Max Left_0-Max];

Right_True = [Max Max];
Right_Out = [-Right_Detect+Max, -Right_Detect+Max];
Right_In = [-Right_0+Max, -Right_0+Max];

figure(8)
hold on
plot(Left_True, Lane, 'k--')
plot(Left_Out, Lane, 'r')
plot(Left_In, Lane, 'r')
plot(Right_True, Lane, 'k--')
plot(Right_Out, Lane, 'r')
plot(Right_In, Lane, 'r')
hold off
title('Lane Departure Warning Evaluation')
legend('True Lane','Lane Detection Range','Location','south')
xlabel('Distance from Center of Lane (m)')
```

References

Ballard, D. B. (1982). *Computer Vision.* Prentice-Hall, Inc.

Chevrolet. (2019). *2019 Chevy Blazer Specifications*. Retrieved April 9, 2019, from https://www.chevrolet.com/suvs/blazer-sporty-suv/build-and-price/features/trims/table?section=Dimensions&styleOne=403396

CSS Electronics. (2019). *CAN bus Explained - A Simple Intro*. Retrieved from https://www.csselectronics.com/screen/page/simple-intro-to-can-bus/language/en#CAN-Bus-Intro-Dummies-Basics

Gans, N. D. (2009). *A hardware in the loop simulation platform for vision-based control of unmanned air vehicles.* Elsevier, Mechatronics (19). University of Texas, University of Florida, and Virginia Polytechnic Institute.

Gruyer, D. G. (2012). *Modeling and validation of a new generic virtual optical sensor for ADAS prototyping.* Institute of Electrical and Electronics Engineers.

Institute of Quality Control. (2018, August 4). *Mobileye Vision Technologies Ltd. Certificate.* Retrieved from https://cdn.mobileye.com/wp-content/uploads/2019/02/ISO-9001.2015-Certificate-2018-EN.pdf

International Organization for Standardization. (n.d.). Retrieved from ISO 9000 family - Quality management: https://www.iso.org/iso-9001-quality-management.html

Ledin, J. (1999). Hardware-in-the-Loop Simulation. In *Embedded Systems Programming* (pp. 42-60).

Litman, T. (2019). *Autonomous Vehicle Implementation Predictions.* Victoria Transport Policy Institute.

Muller, S. H. (2015). *Robustness Evaluation and Improvement for Vision-Based Advanced Driver Assistance Systems.* Institute of Electrical and Electronics Engineers.

Prabowo, Y. T. (2015). *Hardware-in-the-loop simulation for visual servoing of fixed wing UAV.* Institute of Electrical and Electronics Engineers.

Semiconductor Components Industries, LLC. (November, 2018). *1/3-Inch Wide VGA CMOS Digital Image Sensor.* Retrieved from https://www.onsemi.com/PowerSolutions/document/MT9V024-D.PDF

U.S. Department of Transportation. (2018). *Preparing for The Future of Transportation: Automated Vehicles 3.0.*

Ward's Intelligence. (2017). *% Factory Installed Electronics/ADAS Equipment on U.S. Cars and Light Trucks, '16 Model Year.*

Zhang, G. L. (2006). *US Patent No. US7768527B2.*

Zhou, J. S. (2016). *A Framework for Virtual Testing of ADAS.* SAE International.